UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

GENERACIÓN PROCEDURAL DE NIVELES CON AJUSTE DE DIFICULTAD PARA VIDEOJUEGOS DEL GÉNERO DE PLATAFORMAS

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

DANIEL TORRES ROBLEDO

DIRECTORA DE TESIS:

MAT. MARÍA CONCEPCIÓN ANA LUISA SOLÍS GONZÁLEZ COSÍO

CIUDAD DE MÉXICO, 2019

Hoja de Datos del Jurado

1. Datos del alumno	1. Datos del alumno
Apellido paterno	Torres
Apellido materno	Robledo
Nombre(s)	Daniel
Teléfono	55 3956 8584
Universidad Nacional Autónoma de México	Universidad Nacional Autónoma de México
Facultad de Ciencias	Facultad de Ciencias
Carrera	Ciencias de la Computación
Número de cuenta	306041013
2. Datos del tutor	2. Datos del tutor
Grado	Mat
Nombre(s)	María Concepción Ana Luisa
Apellido paterno	Solís
Apellido materno	González Cosío
3. Datos del sinodal 1	3. Datos del sinodal 1
Grado	M. en C
Nombre(s)	María Guadalupe Elena
Apellido paterno	Ibargüengoitia
Apellido materno	González
4. Datos del sinodal 2	4. Datos del sinodal 2
Grado	Dr
Nombre(s)	Verónica Esther
Apellido paterno	Arriola
Apellido materno	Ríos
5. Datos del sinodal 3	5. Datos del sinodal 3
Grado	Dr
Nombre(s)	Jesús
Apellido paterno	Savage
Apellido materno	Carmona
6. Datos del sinodal 4	6. Datos del sinodal 4
Grado	M. en C.
Nombre(s)	Edgar Omar
Apellido paterno	Cebolledo
Apellido materno	Gutiérrez
7. Datos del trabajo escrito	7. Datos del trabajo escrito
Título	Generación procedural de niveles con ajuste de dificultad para videojuegos del género de plataformas
Subtítulo	
Número de páginas	117
Año	2019

Generación procedural de niveles con ajuste de dificultad para videojuegos del género de plataformas

por
Daniel Torres Robledo

Resumen

La Generación Procedural de Contenido (*PCG*, Procedural Content Generation) se refiere a la creación de contenido por medio de algoritmos utilizando datos de entrada limitados o indirectos. El contenido generado para un videojuego pude ser niveles, mapas, reglas de juego, texturas, historias, objetos, música, armas, vehículos, personajes, escenarios, ciudades, etc.

La *PCG* ofrece ventajas como bajo consumo de memoria, disminuir costos de desarrollo, creación de contenido virtualmente ilimitado y personalizado. Sin embargo, se debe analizar como se incluirá esto en el diseño del videojuego, desde que fase y los elementos que se van a generar.

En este trabajo se utiliza la *PCG* para generar niveles para un videojuego del género de plataformas, que ajusta la dificultad del contenido con el fin de lograr un balance con respecto a las habilidades del jugador.

Agradecimientos

Dedico este espacio para agradecer a todas las personas que han formado una parte importante de mi formación académica. En particular quiero agradecer de manera especial a mi tutora María Concepción Ana Luisa Solís González Cosío por su guía y apoyo en la realización de este trabajo, así como sus enseñanzas durante la licenciatura.

También quisiera agradecer a los sinodales María Guadalupe Elena Ibargüengoitia González, Verónica Esther Arriola Ríos, Jesús Savage Carmona y Edgar Omar Cebolledo Gutiérrez por sus observaciones y correcciones para hacer de éste un mejor trabajo.

Finalmente quiero agradecer a la Universidad Nacional Autónoma de México por brindarme la oportunidad de estudiar la licenciatura de Ciencias de la Computación.

Dedicatoria

A mi familia:
Argentina, Ramón y Anaid

A mis amigos:
Hugo, Enrique y Uriel

Gracias a todos por su gran apoyo, sin ustedes esta tesis no hubiera sido concluida.

Índice general

Introducción

El desarrollo de videojuegos requiere de una gran cantidad de recursos, ya sea monetario o de tiempo, debido a esto, se ha buscado crear herramientas que faciliten dichas tareas, con el fin de disminuir costos de producción. Debido a la diversidad de disciplinas que intervienen durante el diseño, existe un área de oportunidad muy amplia para buscar soluciones a diferentes tareas dentro de estos ámbitos.

Una de estas áreas es la generación procedural de contenido, en donde el objetivo es obtener, por medio de un algoritmo, elementos que cumplen con las características de velocidad, confiabilidad, expresividad y controlabilidad, los cuales pueden ser incorporados al videojuego. Dependiendo de la implementación y su inclusión en determinada etapa del desarrollo, es utilizada con diferentes fines.

Esto permite a empresas de desarrollo de videojuegos poder incluir una gran cantidad de contenido en sus productos, a pesar de contar con presupuestos reducidos o con poco tiempo para su desarrollo, siempre que se decida incorporar la *PCG* al proceso de diseño de manera oportuna.

En este trabajo, se propone una solución que permite crear de manera procedural niveles para videojuegos del género de plataformas, además de mostrar su implementación utilizando el *Framework* de *Infinite Mario Bros*, con el objetivo de presentar a los jugadores niveles, que se adapten a sus habilidades y así lograr una balance con respecto a la dificultad de cada uno.

En el Capítulo 1, se define qué es la generación procedural de contenido, que función tiene y las ventajas y desventajas de incorporarse durante desarrollo del videojuego, en qué fase se debe incorporar y algunos ejemplos de su uso en diferentes ámbitos y épocas.

En el Capítulo 2, se describe cómo impacta la relación entre la habilidad de un jugador y la dificultad de un videojuego. También se define el término de *Flujo*, que es un estado al que las personas pueden llegar si la tarea que realizan cumple con ciertos elementos, así como su interpretación dentro del contexto de videojuegos y las características que debe cumplir. También se explican las fases del ciclo iterativo que se debe seguir para ajustar la dificultad del contenido generado.

En el Capítulo 3, muestra las propiedades requeridas por una solución procedural y diferentes aproximaciones al contenido generado de esta manera. Posteriormente se define la construcción

de un nivel del género de plataformas, los elementos que lo componen y la jerarquía utilizada para crear los niveles proceduralmente, así como los datos que recibe y genera cada fase del ciclo iterativo de ajuste pasivo de dificultad.

En el Capítulo 4, se describe la implementación de la solución procedural, en donde primero se utiliza un algoritmo genético para crear la estructura del nivel, así como la codificación de los elementos de la población, población inicial, función de ajuste, y condición de término. Posteriormente se de describe una gramática libre de contexto, que se utiliza para generar todo el contenido del nivel, enemigos, plataformas y recompensas, siguiendo la estructura generada. También se describe cómo el sistema de análisis funciona para generar los elementos a modificar en cada nivel, dependiendo de las acciones del jugador en el nivel anterior.

En el Capítulo 5, se presentan las pruebas realizadas para conocer el desempeño del generador de contenido. Posteriormente se muestran los resultados obtenidos y se analiza cada propiedad requerida por una solución procedural y la opinión de los jugadores con respecto a la dificultad de cada nivel.

Finalmente, en el Capítulo 5.3.2, se presentan las conclusiones del trabajo.

También se incluyen dos apéndices. En el apéndice A se realiza una síntesis de la estructura de un videojuego, lo cual es necesario para poder diseñar un generador de contenido que adapte la dificultad utilizando los elementos de un videojuego. En el apéndice B, se presenta el cuestionario que fue aplicado a los jugadores durante las pruebas de juego.

Capítulo 1

Aplicaciones de la generación procedural de contenido

Se define como Generación Procedural de Contenido (*PCG* por sus siglas en inglés, *Procedural Content Generation*) a la creación de contenido por medio de algoritmos utilizando datos de entrada limitados o indirectos. El tipo de contenido generado pueden ser niveles, mapas, reglas de juego, texturas, historias, objetos, música, armas, vehículos, personajes, escenarios, ciudades, etc[12].

En el contexto de videojuegos, uno de los objetivos de utilizar este tipo de técnicas es poder generar virtualmente una cantidad infinita de contenido. Además de disminuir los tiempos y costos de desarrollo en la creación del contenido de los videojuegos.

1.1 La *PCG* en los videojuegos

La generación procedural de contenido en videojuegos consiste en la automatización utilizando algoritmos, en diferentes grados, para generar elementos que cumplan con ciertas reglas o características definidas en los elementos formales durante el diseño de un videojuego[15].

El contenido procedural puede ser difícil de controlar y en algunos casos resultar no ser mejor que algo desarrollado por un diseñador, además de que la inclusión de esta herramienta agrega un costo monetario y de tiempo al desarrollo del videojuego, por lo que se deben tener presentes las ventajas y complicaciones que pueden surgir derivadas de la decisión de utilizar contenido generado de manera procedural, con el fin de evitar poner en riesgo el completar los avances de cada fase de desarrollo dentro del tiempo definido en el documento de diseño del videojuego.

Por otra parte, el utilizar contenido generado proceduralmente permite obtener resultados inimaginables, niveles infinitos, objetos con características únicas o retos que se adaptan a la habilidad del jugador. De esta manera los jugadores pueden tener su propia aventura y al mismo tiempo suficientes experiencias en común con otros jugadores.[15].

1.2 La importancia de la *PCG* en los videojuegos

Otro punto importante es que, dadas las limitaciones de almacenamiento de las computadoras en la década de los 80, fue necesario el desarrollo de técnicas de generación procedural de contenido para poder guardar solo una parte del juego y crear todo el contenido en tiempo de ejecución cuando fuera requerido.

Un ejemplo de esto es el videojuego *Elite*[1], donde se logró resolver este problema utilizando únicamente una semilla como parámetro de una función, en donde se ejecuta un algoritmo con un número fijo de ciclos para obtener una secuencia de números específicos que determinan la composición de cada planeta (posición en la galaxia, precios, economía, nombre y detalles; los textos son seleccionados numéricamente de una tabla para producir descripciones únicas), en total se obtienen 8 galaxias, cada una con 256 planetas.

Otro ejemplo de videojuego que utilizó *PCG* en la misma época fue *Rogue*[2], un videojuego del género de calabozos, donde los niveles son generados colocando aleatoriamente el primer calabozo y posteriormente se agregan los demás de manera que se pueda tener un camino desde el inicio hasta la meta. Una de las principales ventajas de éste tipo de videojuegos es generar experiencias de juego únicas para los jugadores y utilizar poco espacio en memoria, sin embargo la calidad visual es limitada.

Actualmente, desarrollar un videojuego es un proceso complejo y conlleva una considerable cantidad de tiempo, por lo que si no se planea correctamente desde el inicio, los recursos se terminarán agotando, llevando así a tener un producto incompleto o la cancelación del mismo. Una opción para compensar estas posibles complicaciones es utilizar *PCG* durante el desarrollo, siempre y cuando se decida integrar de manera oportuna y correcta en el diseño, de lo contrario, pueden generarse problemas aún mayores[15].

1.2.1 Cuándo utilizar *PCG*

Se debe tener en cuenta que el incluir contenido generado proceduralmente en un videojuego requiere una inversión de tiempo y desarrollo, ya que aún no existe código que sea totalmente genérico para ser incluido directamente[15].

La clave para crear un módulo de *PCG* es especificar las reglas y procedimientos en el diseño del videojuego, las cuales permitirán mantener un diseño coherente y funcional durante todo el desarrollo.

[1] 2019, https://en.wikipedia.org/wiki/Elite_(video_game)
[2] 2019, https://en.wikipedia.org/wiki/Rogue_(video_game)

Cuando se trabaja de manera individual, estas reglas y procedimientos también servirán como guía para asegurar que el desarrollo se mantenga coherente a lo largo del tiempo, además de permitir experimentar con cambios de manera segura, es decir, sin crear algo totalmente inapropiado o que sea ajeno al contenido del videojuego.

Cuando el modelo de trabajo es en equipo, se enfatiza la importancia de establecer reglas y procedimientos, ya que es importante que todos los miembros tengan en mente la misma idea de cómo se debe de ver y sentir el videojuego, para asegurar que la contribución de cada uno sea funcional.

La generación procedural de contenido puede ser incluida en el videojuego de manera integral en el diseño del videojuego desde el comienzo y como parte fundamental en el núcleo del juego, un ejemplo de esto es el videojuego *No Man's Sky*[3], en donde se genera cada planeta y todo su contenido de manera procedural (hojas, plantas, animales, etc).

Otro uso de la *PCG* es crear una gran cantidad de contenido para posteriormente ser seleccionado y pulido por un diseñador antes de ser incluido dentro del juego. Esta técnica se utiliza generalmente en videojuegos de tipo mundo abierto[4], donde una de sus características es poseer mapas de gran tamaño en donde el jugador puede explorar libremente, como *Skyrim*[5], donde grandes secciones son generadas de manera procedural y después ajustadas por un diseñador. Otro ejemplo es incluir juegos de lógica[6], donde se generan miles de combinaciones jugables y posteriormente se seleccionan las más interesantes por un diseñador. Para esto, se debe tomar la decisión de incluir *PCG* desde las primeras etapas del diseño pero no necesariamente desde el comienzo.

En estos casos el uso de *PCG* inicia y termina pronto en las etapas del desarrollo del videojuego. Una vez que se obtiene el contenido mediante esta técnica se deja de utilizar la herramienta y los diseñadores comienzan a tomar las salidas que más se apegan a lo requerido, para posteriormente ajustarlas de forma manual.

Algunos videojuegos utilizan la *PCG* como un valor agregado, y el núcleo del juego no depende de esto. Se utiliza como una pieza adicional, que generalmente es llamada *Modo infinito*, la cual no forma parte de la historia principal, pero permite al jugador experimentar nuevos retos, niveles y elementos para interactuar[15].

Al ser opciones diferentes, el módulo de *PCG* puede ser creado en una etapa avanzada del desarrollo del videojuego. La generación procedural que copia el núcleo del videojuego ha mostrado requerir menos recursos y ser más fácil de desarrollar, ya que el diseño del videojuego está más avanzado y se tiene una visión más clara de cómo se debe ver y funcionar.

[3] https://en.wikipedia.org/wiki/No_Man's_Sky
[4] https://en.wikipedia.org/wiki/Glossary_of_video_game_terms#open_world
[5] https://en.wikipedia.org/wiki/The_Elder_Scrolls_V:_Skyrim
[6] https://en.wikipedia.org/wiki/Puzzle_video_game

Otra forma de utilizar *PCG* es solo en elementos específicos, como zonas, contenido de escenas, elementos que deben comportarse de diferente manera o seguir reglas específicas, de esta forma el desarrollo no compromete al videojuego, ya que de no ser posible completar el módulo, basta ser generado por un diseñador.

1.2.2 Cuándo no se recomienda el uso de *PCG* en un videojuego

El forzar la inclusión de algoritmos para la generación procedural de contenido puede llegar a poner en riesgo el proyecto, por lo que es importante entender los riesgos al utilizar *PCG* en el diseño del videojuego, así como definir si es necesario incluirse y desde qué etapa se debe comenzar el desarrollo.

Uno de los principales problemas al utilizar *PCG* en videojuegos desarrollados por empresas *AAA*[7], es que el departamento de calidad (*QA* por sus siglas en inglés, *Quality Assurance*), no puede garantizar la calidad al 100% del contenido generado, ya que no es posible tener suficiente personal que pruebe cada salida obtenida, y el módulo de generación procedural de contenido podría mostrar niveles que contengan errores con baja probabilidad[8], los cuales serán vistos por los jugadores hasta que ha sido liberado el producto[15].

En muchas ocasiones se ve el utilizar *PCG* como una forma de ahorrar tiempo de desarrollo, pero no necesariamente es así. Diseñar el generador de contenido puede ser más complejo de lo esperado, principalmente si es un área donde no hay suficiente código que sea confiable o exista suficiente investigación. También es necesario tener en cuenta la cantidad de contenido que se desea obtener, ya que en algunas ocasiones podría ser más viable que sea creado por un diseñador que desarrollar el módulo.

Otro caso en donde no se recomienda utilizar *PCG* es cuando se espera que el videojuego ofrezca el contar una historia de manera estructurada, principalmente en los del género basado en historias, donde se guía al jugador para poder contar todo el guión o que realice tareas específicas definidas por los desarrolladores, y así lograr la experiencia de juego deseada[15].

Cuando el videojuego se desarrolla en ambientes competitivos, como en juegos multijugador[9], por ejemplo *Age of Empires*[10], donde el mapa debe estar balanceado para no otorgar ventaja a alguno de los jugadores, desarrollar un generador de contenido de este tipo llega a ser demasiado difícil, principalmente al garantizar que no se le dará ventaja a alguno de los competidores, ya que una pequeña diferencia puede afectar en gran medida el desarrollo y resultado de la batalla.

[7] Empresas con presupuesto alto para desarrollo y mercadotecnia
[8] Del 0.1%
[9] Un juego que permite a varios jugadores jugar al mismo tiempo.
[10] https://en.wikipedia.org/wiki/Age_of_Empires_II

Como se mencionó anteriormente, se llega a ver la *PCG* como una forma de ahorrar tiempo, pero en algunas ocasiones, debido a esta idea, se termina diseñando un generador totalmente aleatorio, lo cual es una mala idea ya que este contenido llega a ser repetitivo o aburrido, y en los peores casos, no ser jugable.

Por último, cabe mencionar que el término *PCG* se ha vuelto muy popular en el diseño de videojuegos, en donde se utiliza como parte de la publicidad mencionando la cantidad de contenido que puede existir, prácticamente ilimitada. Pero esto no debe ser lo único en lo que se base el videojuego, es necesario tener visualizada la mecánica del juego, así como los elementos básicos del núcleo, de no ser así, incluso el contenido creado por un diseñador no será agradable para el jugador[15].

1.2.3 Ventajas de utilizar *PCG*

Una vez que se han revisado las necesidades del videojuego y se ha decidido el contenido que será generado de manera procedural, la etapa del desarrollo en que se comenzará a crear el módulo y la forma en que se utilizará, es necesario analizar cuáles son las ventajas de utilizar PCG, que pueden ser prácticas o subjetivas.

Algunas de las razones prácticas para generar contenido de manera procedural son[15]:

- **Limitaciones técnicas**: En la época en la que iniciaba el desarrollo de videojuegos, las técnicas de generación procedural de contenido se utilizaron para crear contenido que era imposible guardar en la memoria de una computadora. El videojuego *Frontier: Elite II*[11] contenía una galaxia completa, con estrellas y planetas en un disco flexible utilizando *720 KB* de espacio. Con la capacidad de cómputo de la actualidad ésto ya no es un problema, pero puede ser utilizado en dispositivos electrónicos pequeños u otro *hardware* emergente.
- **Ahorrar tiempo**: Se pueden producir grandes cantidades de contenido más rápido que hacerlo de manera manual. Hay dos opciones para utilizar esta herramienta: generar el contenido y luego ser pulido de manera manual para que se ajuste al videojuego u obtener variedad de contenido (ciudades, vegetación, construcciones, armas, etc.). Hay que tener en cuenta que el desarrollo del módulo de generación procedural de contenido requiere tiempo de desarrollo y pruebas, por lo que no necesariamente se ahorrará tiempo.
- **Expandible**: Permite que cada modificación hecha al generador sea integrada en todo el contenido. Esto significa que todo el trabajo realizado sea multiplicado de manera considerable en todo el videojuego.
- **Rejugable**: Permite que el videojuego contenga virtualmente una cantidad infinita de contenido, ofreciendo al jugador nuevas experiencias de juego.

[11] https://en.wikipedia.org/wiki/Frontier:_Elite_II

- **Código reutilizable**: Con algunas modificaciones, el generador que en un videojuego crea un desierto, en otro puede producir montañas.
- **Aplicación de reglas**: Asumiendo que está libre de errores, permite aplicar rigurosamente reglas para asegurar la conexión de elementos en todo el contenido generado e incluir restricciones en cuanto al balance de dificultad, *puzzles* que tengan solución o niveles que sean jugables y terminables.
- **Modelar la realidad**: En algunas ocasiones se utilizan técnicas de simulación en los generadores procedurales para producir terreno u objetos orgánicos. Los generadores basados en celdas están inspirados en la vida, y pueden ser muy efectivas al producir detalles similares a la realidad. Un ejemplo de esto es *Dwarf Fortress*[12], el cual incorpora una gran cantidad de leyes de la física (o aproximaciones) en su mecánica de juego y generación del contenido.

También existen razones subjetivas para preferir utilizar contenido generado proceduralmente, algunas de las cuales pueden ser[15]:

- **Experiencias únicas**: Cada nivel jugado ofrece diferente contenido, dependiendo del objetivo del generador puede ser un reto nuevo, una pieza de música o un paisaje diferente consistente con el videojuego, produciendo una experiencia personalizada a cada jugador.
- **Diferentes modos de interacción o jugabilidad**: Permite a los usuarios una nueva forma de interacción con el videojuego, que consiste en tratar de predecir patrones obtenidos en cada nivel convirtiendo la herramienta en un elemento más del juego.
- **Impredecible**: Incluso los diseñadores del generador de contenido no pueden predecir la totalidad los elementos generados, lo que les permite disfrutar de su creación de igual manera que un usuario final.
- **Sistemas vivos**: Permite crear sistemas complejos y reactivos como fuego, vegetación, agua, clima, poblaciones, enfermedades, culturas y vida en todos los niveles. Si esto es diseñado por un humano parecería repetitivo comparado con la variabilidad que un generador procedural puede ofrecer. Algunas técnicas basadas en celdas que toman decisiones respecto solo a sus vecinos pueden producir comportamiento emergente[13], el cual puede verse similar a la realidad, ya que en la mayoría de los casos así es como funciona.
- **Creaciones fuera de nuestra imaginación**: El contenido obtenido por un generador procedural puede ser raro, producir contenido que ningún humano pudiera haber imaginado. Las computadoras no piensan de la misma forma en que lo hacen los humanos y pueden llegar a los extremos de la lógica que están fuera de nuestra imaginación.

[12] https://en.wikipedia.org/wiki/Dwarf_Fortress
[13] Ofrecer comportamientos inesperados como resultado de la ejecución del sistema[4]

- **Sensación de infinito**: Definir el concepto de infinito puede resultar complejo, sin embargo un generador de contenido permite tener la sensación de poder jugar un nivel completamente diferente en cada sesión. Un generador procedural de calabozos bien delimitado permite al jugador sentir que domina un conjunto ilimitado de este tipo de niveles, dado que el núcleo del juego se mantiene.
- **Diversión**: Jugar con los parámetros de un generador de contenido puede llegar a ser divertido, con presionar un botón se puede obtener una creación totalmente nueva, y al modificar un número y presionar de nuevo el botón se obtiene algo totalmente diferente.

1.3 El impacto de la *PCG* en los videojuegos

Durante la década de los 80, el principal uso de la *PCG* fue el poder superar las limitaciones técnicas de las computadoras, ya que la memoria y la capacidad de proceso eran muy bajas, lo que permitió obtener gran contenido a partir de guardar solo unos pocos datos y el resto generarlo de manera procedural en tiempo de ejecución.

Cabe mencionar que los fines de utilizar la *PCG* han cambiado con el tiempo, esto es en gran parte debido al aumento de poder de cómputo, actualmente se enfoca en generar una gran cantidad de contenido con menor esfuerzo por parte de los diseñadores, y de esta manera, permitir a pequeñas empresas que desarrollan videojuegos crear productos con gran cantidad de contenido, sin la necesidad de tener presupuestos altos.

Un uso de la *PCG* que se ha mantenido invariante al paso del tiempo es el obtener contenido virtualmente ilimitado permitiendo a los jugadores tener nuevas experiencias por cada nivel jugado, ya sea en mapas, niveles o su contenido.

A continuación se muestran algunos ejemplos de videojuegos que utilizan generación procedural de contenido con diferentes fines, así como una breve descripción, el impacto que tuvo y la época en la que fue desarrollado.

1.3.1 Rogue

Publicado por la compañía *Ai Design* en el año 1980. Es un videojuego del género *RPG*[14] de calabozos, donde en cada salón se encuentran monstruos y tesoros, cuya meta es llegar al nivel más bajo y obtener el *Amuleto de Yendor*, para luego regresar con éste a la superficie para completar el nivel. Este videojuego es considerado uno de los primeros en utilizar técnicas de

[14] *Role-play Video Game*, es un género donde el jugador controla las acciones de uno o varios personajes dentro de un mundo.

generación procedural, cuyo principal fin es obtener una gran cantidad de calabozos y contenido utilizando poca memoria.

La implementación de *PCG* en este videojuego marcó un género de videojuegos conocido como *Roguelike*, cuya principal característica es generar mapas de calabozos. Ésta técnica tiene dos variantes, la primera consiste en colocar aleatoriamente los cuartos y luego unirlos utilizando corredores, mientras que la segunda es colocar corredores aleatoriamente y luego acomodar los cuartos, con el fin de garantizar que habrá un camino para conseguir completar el objetivo.

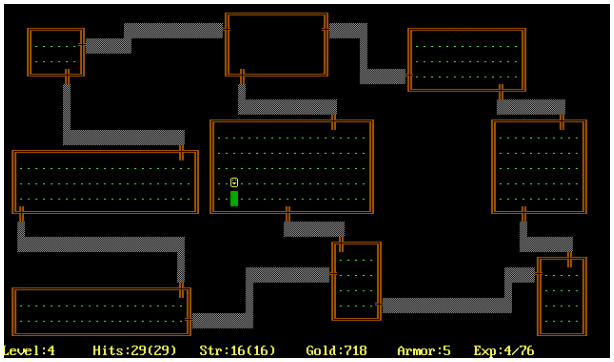


FIGURA 1.1: Mapa de un calabozo generado, al centro se puede observar el *Amuleto de Yendor* (Figura tomada de https://en.wikipedia.org/wiki/Rogue_(video_game)).

1.3.2 Elite

Publicado en el año 1984 por la compañía de videojuegos *Acornsoft*, originalmente para las computadoras *BBC Micro* y *Acorn Electron*. Es un videojuego de mundo abierto con una historia no lineal, en la que el jugador puede completar las misiones en diferente orden, y el uso de gráficas en 3D, utilizando un algoritmo de *wireframe*[15] con imágenes semi-transparentes, hicieron de éste un juego innovador, por lo que fue portado, virtualmente, a toda computadora personal de esa época.

El jugador inicialmente controla al personaje *Commander Jameson*, cuyo nombre se puede cambiar cada ocasión que se guarda el avance de la partida. Inicia con 100 créditos y una nave espacial de poco armamento. La economía se puede incrementar cazando tesoros, completando misiones o llevando a cabo intercambios. El dinero obtenido se puede utilizar para mejorar la nave inicial, ya sea mejorando el armamento, la capacidad de energía, automatizar el aterrizaje, etc.

Debido a la escasa memoria que poseían las computadoras para las que originalmente fue desarrollado, fue necesario utilizar generación procedural para crear el contenido del videojuego,

[15]Representación visual de un objeto 3D utilizado en graficación 3D por computadora.

el cual contiene 8 galaxias con 256 planetas cada una. La técnica utilizada por el algoritmo consistía en guardar solo un número que tiene la función de semilla, y ejecutarlo un número fijo de ciclos para generar una secuencia de números que determina la composición de cada planeta (posición en la galaxia, precios de objetos, nombre y detalles específicos; los cuales se buscan en una tabla y se producen combinaciones únicas). Gracias a esto, no se requiere memoria extra para guardar la información de cada planeta, donde cada uno es único con características fijas. Cabe mencionar que cada galaxia también es generada proceduralmente desde el inicio, originalmente los desarrolladores proponían tener 248 galaxias, pero *Acornsoft* insistió en mantener un universo más pequeño.

FIGURA 1.2: Gráficas del videojuego *Elite* utilizando *wireframes* (Figura tomada de https://en.wikipedia.org/wiki/Elite_(video_game)).

1.3.3 Diablo

Desarrollado por la compañía de videojuegos *Blizzard* a finales del año 1996, cuyas plataformas objetivo fueron *Microsoft Windows*, *Classic Mac OS* y *PlayStation*, los requerimientos mínimos[16] del sistema para poder ejecutarse eran: procesador *Pentium* a *60Mhz* y *8MB RAM*.

Probablemente es el mejor ejemplo del género *action-RPG*[17], en el cual la curva de aprendizaje es baja. Debido a ésto surgió el concepto *action-inspired RPG*, e inspiro a varios juegos, que con el tiempo fueron conocidos como *Diablo clones*[18].

Además de la historia principal, tiene la opción de jugar niveles generados proceduralmente para obtener más puntos de experiencia, subir de nivel, conseguir objetos y enfrentar nuevos retos. Los calabozos son generados de manera aleatoria siguiendo los parámetros definidos por el nivel, para las catacumbas se utilizan corredores largos y cuartos pequeños mientras que las cuevas

[16] https://www.diablowiki.net/Diablo_I
[17] Una combinación de los géneros de acción y *RPG*.
[18] https://www.diabloii.net/blog/comments/top-ten-diablo-clones

siguen un patrón no lineal, después de generar el nivel, los enemigos y objetos son colocados dentro de los cuartos de igual forma.

FIGURA 1.3: Calabozos del videojuego *Diablo* (Figura tomada de https://en.wikipedia.org/wiki/Diablo_(video_game)).

1.3.4 No Man's Sky

Desarrollado por *Hello Games*, un estudio independiente, para las plataformas *PlayStation 4* y *Microsoft Windows*. Fue publicado en agosto del año 2016. Es un videojuego del género *RPG*, que utilizando generación procedural ofrece al jugador 18 trillones[19] de planetas con características únicas generadas de manera determinista para explorar.

Durante el desarrollo del videojuego, el autor buscó capturar la idea de exploración del espacio y el optimismo descritos en la literatura de ciencia ficción de las décadas de los 70's y 80's.

El videojuego consiste en obtener recursos para mejorar la nave espacial, herramientas y el traje espacial. Estos se obtienen por medio del comercio, documentando formas de vida y detalles de los planetas visitados.

Cada parte del universo de este videojuego es creada de manera procedural, estrellas, planetas, flora, fauna y encuentros de combate, de manera determinista. Los valores utilizados como semilla para obtener éste contenido son guardados en los servidores del videojuego, y se genera cuando el jugador se acerca a alguna zona, de esta manera se asegura que se mostrarán los mismos elementos a todos los jugadores.

Debido a que casi todo el contenido es generado de manera procedural, llega a ser repetitivo, por lo que recibió malas calificaciones por parte de los jugadores.

[19] 18^{30} posibles planetas

FIGURA 1.4: Vista de un planeta generado por el videojuego *No Man's Sky* (Figura tomada de http://bestgamewallpapers.com/no-mans-sky/new-eridu).

1.3.5 *SpeedTree*

Desarrollado por la empresa *Interactive Data Virtualization* en el año 2003, cuya finalidad es generar una gran cantidad de vegetación siguiendo un patrón para simular variaciones de la misma especie de la planta a generar.

Utiliza algoritmos de generación procedural para simular variaciones de un modelo, lo que permite que el tiempo y costo de diseño se reduzca, ya que solo basta definir la estructura y pequeños detalles como las hojas. Después de configurar estos parámetros, ya sea un árbol completo o solo una parte, el algoritmo generará salidas basadas en el original pero con pequeñas diferencias. Ésta es una forma de simular como se comporta la vida en el mundo real.

Debido al gran impacto que tiene al minimizar costos y tiempos, éste algoritmo se ha incluido en videojuegos como *Saint's Row IV* y películas como *Iron Man 3* y *Thor: Un mundo oscuro*. Actualmente se encuentra disponible para ser incluido en software de desarrollo de videojuegos como *UNITY3D* y *Unreal Engine 4*.

1.3.6 Generación procedural de texturas

Uno de los principales retos en el área de graficación por computadora ha sido el agregar de manera eficiente gran cantidad de detalles visuales a imágenes sintéticas. Hasta ahora, el ruido procedural ha logrado generar este tipo de resultados de manera exitosa.

Desde la primera imagen generada con esta herramienta (Figura 1.6), desarrollada por K. Perlin, el uso del *Perlin Noise* ha sido objeto de estudio por la industria y en el ámbito de la investigación. Este tipo de ruido se utiliza en un amplio y diverso rango de propósitos de generación procedural de texturas como nubes, olas, tornados, trayectorias de cohetes, ondas de calor y movimientos de personajes animados, entre otras[9].

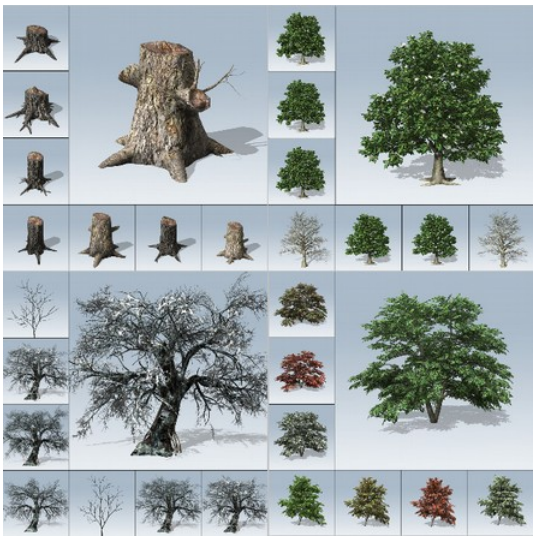
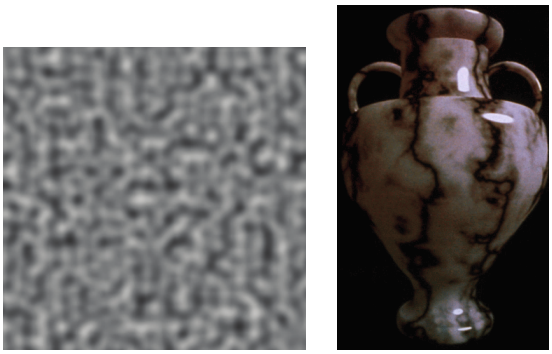


FIGURA 1.5: Diferentes modelos de árboles creados a partir de un original con pequeñas variaciones en los parámetros (Figura tomada de https://store.speedtree.com).



a b

FIGURA 1.6: a: primera función de ruido. b: Textura de una vasija creada de manera procedural utilizando *Perlin Noise* (Figura tomada de [9]).

Esta herramienta actualmente esta implementada en la mayoría de los programas para graficación en 3D como *Autodesk*, *3ds Max*, *Maya*, *Blender*, *RenderMan*, etc.

Una de las principales ventajas de utilizar *Perlin* es la velocidad para evaluarse y su bajo costo de memoria, permitiendo utilizarse en tiempo real para generar detalles visuales de alta calidad en videojuegos. El algoritmo de esta herramienta es altamente paralelizable, lo que permite mejorar aún más los tiempos de evaluación utilizando los múltiples núcleos del procesador y tarjetas gráficas.

Capítulo 2

Relación entre dificultad y la habilidad de un jugador

A mediados de la década de los 70's, el profesor de psicología Mihaly Csikszentmihalyi introdujo el concepto de *Flujo*, el cual se define como la sensación de enfocarse completamente en una actividad con un alto nivel de satisfacción. Para esto desarrolló una serie de teorías que permitieran llevar a las personas a un estado de *Flujo*, las cuales han sido aplicadas en diversos campos para diseñar mejores experiencias[1]. Uno de los logros más importantes de estas teorías es la definición de la *Flow Zone*, conocida también como *La Zona* (*The Zone*) en el contexto de videojuegos[2].

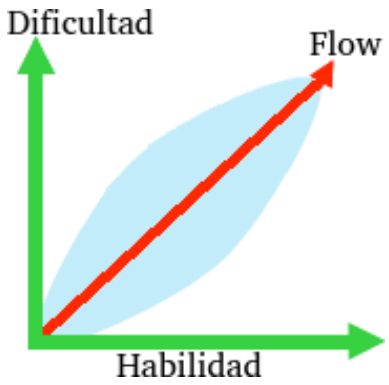


FIGURA 2.1: Zona segura generada por la tolerancia entre la ansiedad y el aburrimiento (Figura adaptada de [2]).

Para mantener a una persona en *La Zona*, la actividad desempeñada debe tener un balance entre reto y sus habilidades. Si el reto es mayor que la habilidad, la actividad llega a ser tan complicada que genera ansiedad, por otro lado, si el reto es menor a la habilidad provoca aburrimiento. Adicionalmente es necesario considerar la tolerancia que el participante tiene, por lo que hay un área segura donde la actividad no es muy difícil o muy fácil, y la ansiedad o el aburrimiento no se presentan[1].

[1] Esto también puede ser aplicado al diseño de cursos.

La descripción del *Flujo* es igual a lo que un jugador experimenta cuando está totalmente enfocado en un videojuego. Durante el tiempo que se mantenga en este estado, el jugador puede perder el paso del tiempo y olvidarse de todas las presiones externas[2].

2.1 Elementos del *Flujo*

De acuerdo a *Csikszentmihalyi*, para poder experimentar un estado de *Flujo* es necesario cumplir con ocho principales componentes[1]:

- Un reto que requiera de habilidades
- La mezcla de acción y conciencia
- Metas precisas y claras
- Retroalimentación continua e inmediata
- Sensación de control
- Concentración en la tarea
- Pérdida de la auto conciencia
- Transformación del tiempo

Tomando el *Flujo* con una perspectiva de diseño de videojuegos, se pueden observar tres elementos clave que se deben tener para lograr experimentar esta sensación:

- Una premisa
- Resultados
- La correcta cantidad de retos de acuerdo a la habilidad del jugador
- Sensación de control en el juego

El lograr integrar estos elementos de manera correcta en el diseño del videojuego, permitirá que el jugador llegue a este estado.

Para permitir que el juego alcance a diferentes tipos de poblaciones, se deben mantener estos cuatro elementos, especialmente para ajustar el reto basado en la habilidad de cada jugador.

Tomando el videojuego como un sistema, el *Flujo* permite explicar la razón por la cual las personas prefieren ciertos juegos sobre otros y cómo se genera una adicción a éstos. Si el juego cumple con los elementos principales del *Flujo*, cualquier contenido puede ser satisfactorio y cualquier premisa ser atractiva[1].

[2] Holt, 2000

2.2 Ajustes del videojuego

Asumiendo que el contenido del videojuego es atractivo para la audiencia, al diseñar un videojuego se debe tener en cuenta cómo mantener al jugador en un estado de *Flujo* y eventualmente terminar el juego. Esto significa que el sistema necesita mantener la experiencia de cada jugador en *La Zona*[1].

En la figura 2.2, la curva roja representa la experiencia obtenida por un jugador durante una sesión de juego. Se pueden observar las transiciones entre aumento y disminución de la dificultad, pero se mantiene en *La Zona* gracias a la tolerancia que los humanos poseen ante la ansiedad y el aburrimiento.

Si los parámetros de dificultad sufren un cambio mayor que los observados en la figura 2.2, la experiencia del jugador saldrá de la zona segura y experimentará ansiedad si el reto no es adecuado para sus habilidades, por lo que romperá el estado de *Flujo*, como es mostrado en la figura 2.3.

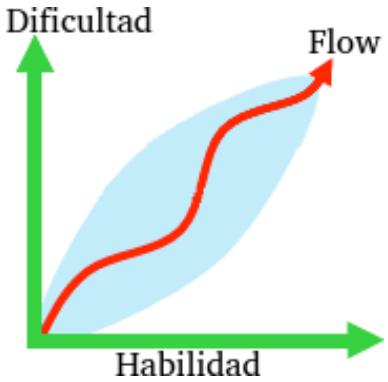


FIGURA 2.2: Jugador experimentando el estado de *Flujo* durante una sesión de juego (Figura adaptada de [2]).

FIGURA 2.3: Momento en que el un jugador sale de la zona de *Flujo* debido a que los retos no corresponden a sus habilidades (Figura adaptada de [2]).

También es importante considerar que las personas tienen diferentes habilidades, así como niveles de tolerancia, por lo que diseñar un juego que mantenga a una cierta población de jugadores con habilidades dentro de la media en un estado de *Flujo*, no funcionará igual para jugadores avanzados o novatos, como se muestra en la figura 2.4. Por esta razón es necesario que la experiencia de juego no sea estática o lineal.

FIGURA 2.4: Diferentes áreas seguras de las zonas de *Flujo* para distintos tipos de jugadores (Figura adaptada de [2]).

Durante el proceso de diseño, se deben realizar iteraciones de prueba, evaluación de resultados y modificación del juego continuamente para lograr la experiencia de juego deseada y llevar al jugador a un estado de *Flujo*. En las pruebas de juego (*playtest*) se utilizan jugadores que representen diferentes poblaciones objetivo para definir las zonas del *Flujo*. Con el fin de obtener experiencias de juego óptimas para una amplia variedad de jugadores, además de tener un área segura de la zona de *Flujo* más grande, es necesario que el sistema pueda adaptarse y unir las experiencias de juego en una, ajustando el *Flujo* basado en las habilidades del jugador.

2.2.1 Ajuste pasivo de la dificultad

Este tipo de ajuste a la experiencia del juego para mantener al jugador en un estado de *Flujo* inicia utilizando datos obtenidos del jugador durante el juego de un nivel, y posteriormente generar contenido nuevo ajustando la dificultad a las habilidades mostradas[1].

Este tipo de sistemas que ajustan la dificultad basados en el desempeño mostrado por el jugador trabajan a través de un ciclo de ajuste iterativo, que consiste principalmente de cuatro elementos:

- **Jugador:** Genera datos crudos dentro del juego durante una sesión.
- **Sistema de control:** Elije los datos que reflejan el estado de *Flujo* del jugador.
- **Sistema de análisis:** Analiza el estado de *Flujo* y notifica al sistema del juego de los cambios necesarios.
- **Sistema del juego:** Aplica los cambios para generar un nuevo nivel.

Teóricamente, este sistema debería de poder mantener al jugador en la zona de *Flujo* ajustando constantemente el nivel del juego utilizando los datos obtenidos del jugador[3]. Sin embargo existen algunos problemas clave que aún no se han resuelto, lo que dificulta la implementación de sistemas pasivos de ajuste[1].

- **Datos indirectos:** Los videojuegos no reflejan lo que el jugador está pensando. Actualmente, la mayoría de las conexiones entre jugadores y videojuegos es a través de los controles. Con una cantidad limitada de datos de entrada, la posibilidad de percibir si el jugador está en un estado de *Flujo* directamente es baja. A pesar de que ya existe *hardware* de biometría y investigaciones sobre el tema, aún existe un área de oportunidad en el campo de registrar emociones. La mayoría de las medidas aún están basadas en suposiciones y estadísticas incompletas.
- **Desempeño no refleja el *Flujo*:** Existen varias formas de estimar el desempeño del jugador a través de muestras limitadas de los datos como precisión, disparos a la cabeza, total de bajas. Las medidas de desempeño son objetivas mientras que el *Flujo* es subjetivo. Cuando un jugador se encuentra en un estado de *Flujo* o solo realizando saltos sin terminar el nivel, el sistema de ajuste tendrá dificultades para determinar eso.
- **Análisis basado en suposiciones:** El hecho de asumir algunas cosas nunca funcionará para todas las poblaciones. Cuando un jugador disfruta realizar una acrobacia suicida en *Grand Theft Auto*[4], sería un error que el sistema asuma que las habilidades del jugador son bajas debido al conteo de muertes.
- **Cambios basados en un diseño rígido:** La forma en que el sistema ajusta la dificultad es definida por el diseñador, quién decide los cambios que serán realizados y cómo se aplican, sin embargo las preferencias de una persona nunca representarán a diferentes poblaciones[5].

2.2.2 Ajuste activo de la dificultad

Considerando los elementos del *Flujo*, los diseños se han centrado en balancear los retos con las habilidades del jugador, sin embargo, se ha ignorado otro elemento importante, que es el permitir que el jugador tenga la sensación de control sobre la actividad que realiza[1].

Comúnmente en los medios de comunicación, la sensación de control proviene de el avance y la retroalimentación positiva[6]. En los videojuegos, el jugador participa de manera activa, por lo que también entra en juego la posibilidad de realizar elecciones que tengan impacto en la experiencia de juego.

[3] Bailey & Katchabaw 2005
[4] https://www.rockstargames.com/V/
[5] Costikyan 2004
[6] Adams 2002

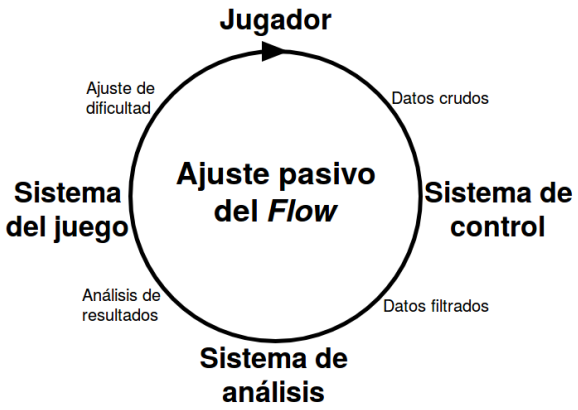


FIGURA 2.5: Diagrama de ajuste de la dificultad de forma pasiva (Figura adaptada de [1]).

Para lograr crear un juego con este tipo de ajuste, es necesario tener una gran cantidad de actividades con diferentes dificultades para alcanzar a diferentes tipos de jugadores. Basándose en los gustos de cada jugador, cada uno tomará diferentes elecciones y avanzará de manera distinta a través del juego.

Una vez que se tiene trazada la gráfica de las actividades que se darán al jugador basadas en sus elecciones, la experiencia del *Flujo* podrá ser ajustada fácilmente por el jugador (figura 2.6), permitiendo que si comienza a sentir aburrimiento elija un nivel de juego con mayor dificultad.

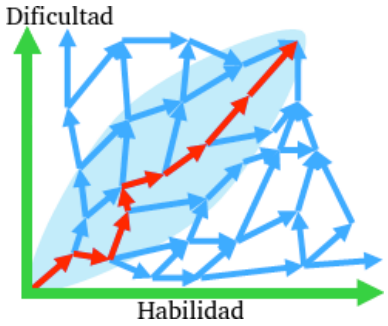


FIGURA 2.6: Diagrama de los puntos de elección para el jugador (Figura adaptada de [2]).

2.2.3 Integración de las elecciones dentro de la mecánica del juego

Con el fin de ajustar las experiencias del *Flujo* dinámicamente y evitar romper con la continuidad del mismo, la posibilidad de tomar una lección debe aparecer con alta frecuencia. Sin embargo, estos ajustes también podrían ser potenciales interruptores para aquellos jugadores que se encuentran en *La Zona*[1].

Una solución a este problema es implementar un sistema de control que detecte cuándo es un momento adecuado para ofrecer elecciones al jugador. Sin embargo, los sistemas de control aún no son lo suficientemente robustos para ser capaces de detectar cuándo un jugador esta dentro de *La Zona*. La solución en la actualidad es integrar las elecciones como parte de la mecánica del juego, permitiendo que el jugador las vea como parte del juego y eventualmente pasen desapercibidas.

Para ofrecer una experiencia interactiva agradable, cuyo objetivo sea alcanzar una gran variedad de jugadores con diferentes habilidades, el diseño del videojuego debe incluir los siguientes cuatro pasos[2]

- Mezclar y unir los componentes del *Flujo*.
- Mantener la experiencia de juego dentro de *La Zona*.
- Ofrecer elecciones a los jugadores para dar la sensación de control.
- Integrar las elecciones dentro del núcleo del videojuego para asegurar la continuidad del *Flujo*.

Capítulo 3

Generación procedural de niveles con ajuste de dificultad

La generación procedural de contenido ha sido parte de los videojuegos desde la década de los 80's, sin embargo, actualmente es un área de estudio el como resolver los problemas que surgen al utilizar esta técnica. A partir de esto, se han desarrollado nuevos métodos y variaciones o combinaciones de las técnicas ya existentes, las cuales se han adaptado para utilizarse en juegos modernos con diferentes propósitos.

Actualmente se han definido tres áreas de la *PCG* que no se pueden realizar con la tecnología actual y posiblemente no sea posible lograrse como se describe[1]. Sin embargo sirven para mostrar las limitaciones y, por consiguiente enfocar esfuerzos de investigación y desarrollo[13].

- **Multi-nivel, multi-contenido:** Se refiere a un generador de contenido que, dada una mecánica de juego y un conjunto de reglas, pueda crear el universo del juego, uniendo los elementos de manera coherente.
- **Diseño basado en *PCG*:** En este caso el generador de contenido es parte fundamental del diseño del videojuego, sin el cual el sistema no podría funcionar de manera adecuada.
- **Generar juegos completos:** Este tipo de generador permite, además de crear contenido, crear el juego completo, es decir, niveles, personajes, reglas, objetos, etc.

Aún existen amplias áreas de oportunidad en el ámbito de *PCG*, sin embargo, es importante tener en cuenta que sí es posible desarrollar generadores para tareas más específicas.

3.1 Propiedades requeridas para una solución de *PCG*

Se pueden definir las implementaciones de *PCG* como soluciones a los problemas de generación de contenido. Las propiedades requeridas, o deseables, para una solución varían dependiendo

[1] Similar a las listas de *problemas no resueltos* en las áreas de Física o Matemáticas

de cada aplicación, sin embargo, lo mínimo que deben cumplir es generalmente un balance entre velocidad y calidad o variabilidad y confiabilidad[13].

Velocidad

Esta característica varía dependiendo de el contexto en que se realiza la generación del contenido, en el caso de que sea durante el juego debe ser lo más rápido posible, por otro lado si es durante el desarrollo del juego es viable que tome más tiempo.

Confiabilidad

La confiabilidad del contenido generado depende del contexto así como de la calidad esperada, en algunos casos una falla en el contenido puede llevar a resultados catastróficos, como crear un nivel de calabozos que no tenga salida, sin embargo al crear objetos como una planta, un error hará que tenga forma extraña sin afectar la mecánica del juego.

Controlabilidad

En algunos casos es necesario poder controlar algunos aspectos del contenido generado, ya sea por el usuario o un algoritmo que se adapte a las habilidades del jugador, esto permite dar la sensación de control al jugador así como mantener la incertidumbre del resultado.

Expresividad y diversidad

Una de las utilidades de la *PCG* es generar variaciones de un elemento dado. Por un lado, un nivel de expresividad nula creará siempre el mismo nivel con cambios aleatorios del color de un solo elemento, en el otro extremo se puede obtener una sensación de un nivel sin sentido e imposible de jugar, en donde se unen los elementos de manera aleatoria.

3.2 Taxonomía de *PCG*

Dada la variedad de métodos así como objetivos y contextos en donde se emplea el contenido generado de manera procedural, es importante observar las diferencias y similitudes de manera general.

Online versus offline

Las técnicas *PCG* pueden ser utilizadas de manera *online*, esto es al mismo tiempo que el jugador está jugando un nivel, lo que permite crear una gran cantidad de variaciones, y dando la sensación de que el juego sea virtualmente infinito así como la oportunidad de crear contenido adaptado a la experiencia del jugador. En el caso de ser un generador *offline* donde el contenido se genera en etapas de desarrollo del videojuego o antes de iniciar una sesión, esta técnica se utiliza principalmente en contenido más complejo como mapas, ambientes o niveles.

Necesario *versus* opcional

La *PCG* puede ser utilizada para generar contenido necesario para completar un nivel, o para generar contenido auxiliar que puede ser descartado o reemplazado. La diferencia principal es que el contenido necesario debe ser correcto para que funcione la mecánica del juego, mientas que en el opcional no aplica esta condición.

Grado de control

La generación de contenido puede ser controlada en diferentes grados. Utilizar una semilla aleatoria para obtener el control de la generación del espacio o utilizar parámetros para controlar aspectos de los objetos a crear.

Genérico *versus* adaptativo

La generación de contenido genérico crea objetos sin tener en cuenta el comportamiento del jugador, en contraste con el adaptativo, en el cual, se crea una experiencia de juego personalizada para cada jugador, basándose en las habilidades y elecciones tomadas para posteriormente analizar dicha información con un sistema y adecuar los parámetros que permitirán generar nuevo contenido.

Estocástico *versus* determinístico

Cuando el contenido es generado de manera determinista, es posible recrear el mismo contenido al usar parámetros iguales, mientras que en un generador estocástico es improbable obtener el mismo resultado.

Constructivo *versus* generación y prueba

En la *PCG* constructiva, el contenido es generado y no es posible realizar modificaciones durante esa sesión. Por otra parte en el contexto de generación y prueba, se deben realizar pruebas hasta que el contenido cumpla con las características deseadas.

Generación automática *versus* mixta

La *PCG* permite una interacción limitada o nula por parte del usuario, ya sea para modificar los parámetros del algoritmo o indirectamente a través del juego, sin embargo el principal propósito es generar virtualmente ilimitadas variaciones de los elementos del juego. Sin embargo, en la generación mixta, el diseñador trabaja junto con el generador de contenido para obtener el resultado esperado, ya sea para completar las partes faltantes o mantener la coherencia del producto.

3.3 Elementos de los niveles en juegos de plataformas

La dificultad principal de la generación procedural de niveles en un videojuego del género de plataformas, es que un pequeño cambio, como la distancia entre dos plataformas, puede impactar en el nivel completo, pasando de ser un reto a ser físicamente imposible de terminar. El no respetar las restricciones o colocar los elementos de manera aleatoria fácilmente puede generar contenido en el que ganar sea imposible.

Las acciones con ritmo facilitan al jugador a llegar a un estado de *Flujo*[2]. El colocar los obstáculos con cierto ritmo crea una secuencia rítmica de movimientos para el jugador, lo que facilita realizar cada acción en el momento adecuado, facilitando los cálculos de distancia y tiempo[3].

Adicionalmente, el utilizar ritmos permite generar niveles más largos utilizando pocos elementos, por ejemplo, en el videojuego *Mario Bros*, al repetir y reorganizar elementos básicos como tuberías, bloques, plataformas y enemigos permite a los diseñadores crear diferentes niveles con experiencias de juego variadas.

Los elementos más básicos de un nivel del juego *Infinite Mario Bros* se denominan componentes, los cuales son plataformas, enemigos, bloques, objetos y *powerUp's*. Estos, a su vez se agrupan en patrones o grupos de ritmos para formar bloques del nivel en donde el jugador debe realizar acciones definidas.

Posteriormente, se utiliza una estructura de cápsulas, las cuales contienen, cada una, un ritmo, y determinan a que cápsulas es posible acceder a partir desde donde se encuentra el personaje. Esto permite obtener contenido con mayor variación y evitar niveles lineales.

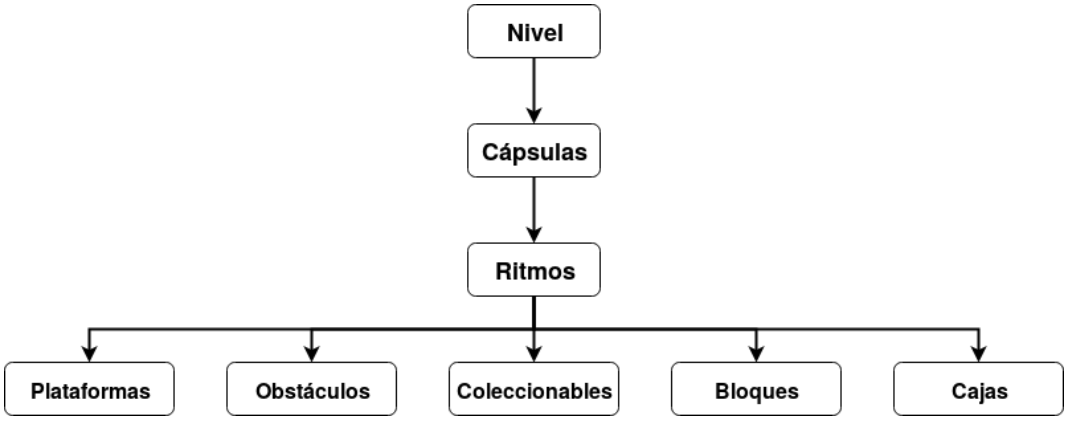


FIGURA 3.1: Jerarquía de los elementos que componen un nivel del videojuego *Infinite Mario Bros*.

[2]Mihaly Csikszentmihalyi - 1990

3.3.1 Componentes

Los componentes como plataformas, bloques y tuberías son los elementos básicos para la construcción de un juego de plataformas. Generalmente un componente tiene un obstáculo y un área de descanso, que puede ser un espacio vacío el cual requiere que el jugador realice un salto seguido de una plataforma en la cual aterrice.

La representación visual de los componentes permite al jugador diferenciar las propiedades de cada uno, tales como fricción, rebote o potencial de daño, y así poder planear sus acciones. Su construcción esta definida por las dimensiones que ocupa en el espacio y el tipo de componente.

Estos componentes se categorizan de acuerdo al rol que tienen, cada uno pude pertenecer a más de una categoría, por ejemplo, las cajas que contienen objetos en *Inifinite Mario Bros* pueden ser consideradas adicionalmente como plataformas[17]. Las categorías de componentes son:

- **Plataformas:** Es cualquier superficie en la que el personaje puede caminar o correr de manera segura. Poseen propiedades físicas como coeficiente de fricción, longitud e inclinación. Estas pueden ser fijas, estar en movimiento o desvanecerse intermitentemente, y sirven para formar rutas que permiten al jugador avanzar para llegar a la meta.
- **Obstáculos:** Es cualquier construcción en el nivel que causa daño al personaje o impide su avance. Pueden ser superados, ya sea eliminándolos o evitándolos con brincos.
- **Ayuda de movimiento:** Son elementos que, cuando el personaje interactúa con ellos, le permiten realizar movimientos adicionales para poder avanzar en el nivel. Ejemplos de estos elementos son escaleras, trampolines o cuerdas, que modifican de manera temporal la dirección, velocidad o capacidad de salto.
- **Objetos coleccionables:** Son recompensas dadas al jugador, que pueden ser monedas, vidas extra, mejoras o armas. También puede ser en forma de puntos que, posteriormente es posible cambiar por un objeto. Otra función de estos elementos es guiar al jugador a través del nivel, mostrando el camino a seguir o la forma y ritmo de los saltos.
- **Disparadores:** Son objetos interactivos que permiten cambiar el estado del sistema. También pueden ser utilizados para crear obstáculos, en los que el objetivo es desbloquear el camino para poder avanzar o conseguir alguna recompensa.

3.3.2 Patrones

Los patrones permiten agrupar componentes individuales en secuencias de mayor longitud manteniendo las acciones del jugador con ritmo[12]. Se proponen 4 tipos de patrones:

1. **Básico:** Consiste en un componente repetido una o varias veces sin variación.

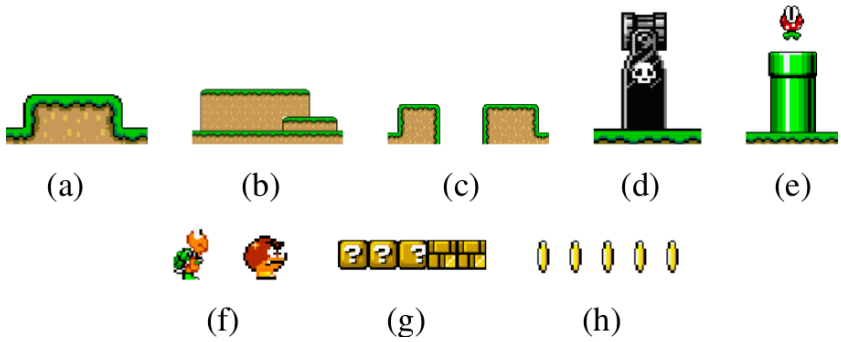


FIGURA 3.2: Representación de los componentes utilizados: (a) plataforma, (b) colina, (c) vacío, (d) cañón, (e) tubería, (f) enemigos, (g) cajas y (h) monedas.

2. **Complejo:** Consiste en repetir el mismo componente, pero con algunos cambios de acuerdo a una secuencia dada, como una serie de saltos de diferente longitud.
3. **Combinado:** Alterna entre dos patrones básicos de diferentes componentes. Un ejemplo de esto es una serie de tres saltos horizontales seguidos por dos obstáculos de espinas.
4. **Compuesto:** Un patrón compuesto consiste en colocar dos componentes que requieren una acción diferente, de tal manera que sea necesario coordinar las acciones del jugador para superar el obstáculo. Un enemigo en movimiento requiere que el jugador tenga en cuenta el tiempo para evadirlo, mientras que es necesario calcular la distancia para brincar un espacio vacío, al estar en conjunto requieren mayor habilidad por parte del usuario.

Con estos patrones solo es posible generar secuencias lineales, las cuales pueden presentar retos adecuados para el jugador, sin embargo, es importante considerar estructuras no lineales como caminos paralelos, ciclos, regresos y niveles secretos para dar la sensación de control en la elección de la ruta para completar el nivel y obtener mayor variedad en el contenido generado.

3.3.3 Ritmos

Los ritmos y descansos son elementos clave para ofrecer al jugador una experiencia de juego agradable, además de tener impacto directamente en la dificultad de los juegos del género de plataformas al incluir retos basados en precisión. Con el fin de utilizar una estructura basada en ritmos para generar el nivel, es necesario encapsular los patrones en grupos de ritmos, los cuales contengan un reto. Cada grupo consiste en un conjunto de acciones que debe realizar el jugador y el patrón que genere los elementos necesarios[17].

Los ritmos son conjuntos de elementos definidos dentro del juego, que al unirse generan un reto que debe superar el jugador, y la gramática utilizada para colocar estos elementos dentro de cada nivel. Estos ritmos poseen dos propiedades tipo y densidad, para la primera pueden ser regulares (saltos a distancias iguales), oscilatorios (alterna saltos largos seguidos de saltos cortos)

y aleatoria (saltos a distancias aleatorias), mientras que la densidad se refiere a la cantidad de saltos que debe haber por ritmo, entre más alto sea el valor de la densidad, más precipicios habrá.

Las acciones que realiza el jugador para avanzar en el nivel en conjunto con el tiempo ideal para presionar los botones esta dado por el ritmo, el cual es definido por el diseñador. En la figura 3.3 se pueden observar dos grupos de ritmos. En el de la izquierda el jugador debe realizar dos acciones idénticas: brincar para conseguir las monedas y eliminar a los enemigos. Mientras que el grupo de ritmos de la derecha es una versión similar más corta, en la que la acción es igual al grupo de la izquierda.

A pesar de que son grupos similares, es importante remarcar la diferencia de los elementos colocados en cada uno, mientras que las acciones del primer grupo se espera que sean realizadas por el jugador de manera continua, en el grupo de la derecha deberá tomar un tiempo para coordinar los siguientes movimientos. Esta pausa delimita el ritmo y debe estar marcada por una plataforma en la que no se debe realizar alguna acción.

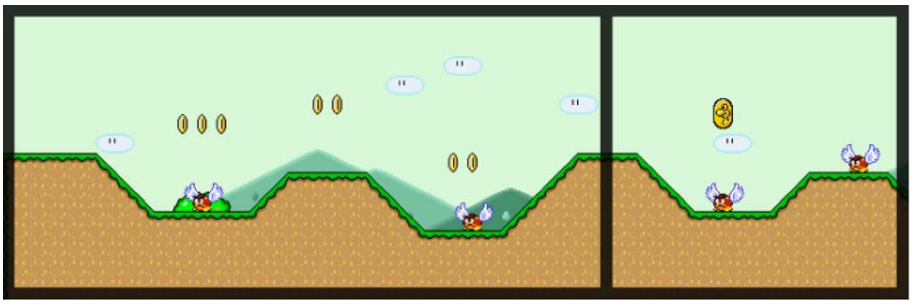


FIGURA 3.3: Dos grupos de ritmos, delimitados por el rectángulo negro. Los grupos de ritmos están separados por una plataforma en donde no se requiere alguna acción (Figura adaptada de [17]).

3.3.4 Cápsulas

Una cápsula está formada por algún patrón o grupo de ritmos, sin embargo, en este nivel de la jerarquía no es necesario considerar las características del mismo, solo se requiere que el jugador pueda acceder del final de un patrón al inicio del siguiente[3].

El generar un nivel utilizando cápsulas permite crear estructuras con diferentes grados de linealidad. En videojuegos del género de plataformas es posible encontrar varios tipos de estructuras, las cuales al unirlas de diferentes formas pueden generar una gran variedad de niveles, cada uno con sus respectivos retos y dificultades.

Entre las estructuras de cápsulas que son compartidas en videojuegos de este género, podemos encontrar ramificaciones, que permiten elegir qué camino tomar para avanzar a la meta, cada una de estas trayectorias ofrece diferentes retos y recompensas, además de ofrecer la sensación de control al jugador.

En la figura 3.4 se muestran las estructuras de cápsulas que serán utilizadas en la implementación del generador de niveles para el videojuego *Infinite Mario Bros*:

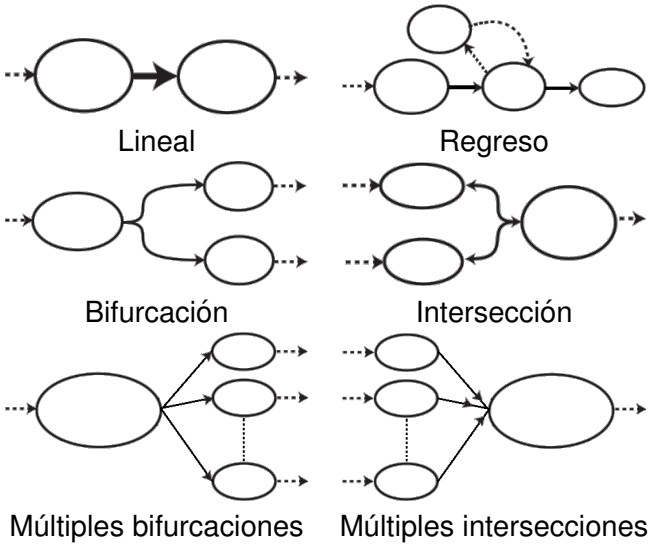


FIGURA 3.4: Estructuras para crear niveles no lineales (Figura adaptada de [3]).

3.4 Ajuste de la dificultad en *Infinite Mario Bros*

Con el fin de ajustar la dificultad de los niveles a generar en el videojuego *Infinite Mario Bros* para mantener al jugador en un estado de *Flujo*, se propone utilizar un sistema de ajuste pasivo, en el cual se genera el nivel a partir de los datos obtenidos de su desempeño en niveles anteriormente terminados.

El ajustar la dificultad permite abarcar diferentes tipos de jugadores, desde los casuales hasta los más competitivos, ya que, debido al aprendizaje durante el juego, las habilidades del jugador tienden a aumentar, por lo que los retos también lo harán.

Como se mencionó en el capítulo 2, el sistema de ajuste pasivo del *Flujo* consiste en un ciclo iterativo de cuatro fases: datos generados por el jugador, sistema de control, sistema de análisis y sistema del juego. Los cuales se describen a continuación, con un enfoque dirigido al videojuego *Infinite Mario Bros*, detallando los procesos y los datos que requiere y genera cada fase.

3.4.1 Información del jugador

En esta fase se obtienen datos a partir del desempeño del jugador durante un nivel. En el caso del videojuego *Infinite Mario Bros*, la única interacción entre el sistema y el el jugador está dada a través de los controles, por lo que el identificar el estado de *Flujo* estará limitada dichos datos.

Los elementos que intervienen en este sistema se describen a continuación, los cuales permiten recabar la información del jugador durante un nivel:

- **Objetivo:** Consiste en llegar a la meta que se encuentra al final del nivel.
- **Obstáculos:** Enemigos, caídas de las plataformas y el tiempo para completar el nivel.
- **Recursos:** *PowerUp'*s para modificar el estado del personaje, así como monedas para conseguir puntos. Por otra parte, el tiempo para terminar el nivel siempre es fijo así como la cantidad de vidas con las que inicia el jugador.
- **Procedimientos:** Los acciones del personaje son avanzar, retroceder, brincar, disparar, correr y agacharse. Solo se puede eliminar a un enemigo brincado encima del mismo, lo cual recompensará con puntos adicionales al jugador. Por parte de los procedimientos que son ejecutados de manera interna en el sistema se encuentran los que mantienen el estado del personaje y el avance del tiempo.
- **Reglas:** Afectan las acciones que puede realizar el jugador dependiendo el estado del personaje, solo puede agacharse cuando se encuentra en un estado diferente a pequeño, y solo puede disparar cuando se encuentra en estado de fuego. Cuando el jugador obtiene un *powerUp* de hongo pasará de estado pequeño a grande, y cuando obtiene uno de flor pasará a estado de fuego, así mismo cuando sea dañado por un enemigo pasará de estado de fuego a grande y finalmente a pequeño, cuando se encuentre en estado pequeño y reciba daño, el jugador pierde una vida y debe comenzar de nuevo el nivel. Otro dato a considerar es que cuando se acabe el tiempo dado y no se haya concluido el nivel, también perderá una vida independientemente del estado en que se encuentre el personaje. Cuando el número de vidas llegue a cero, perderá el juego. Se consigue una vida extra al coleccionar 100 monedas.
- **Reto:** Esto consiste en terminar el nivel evitando los obstáculos y llegando a la meta antes de que el tiempo se acabe. Sin embargo, es importante tener en mente la cantidad de puntos que se obtienen durante el nivel eliminando enemigos y consiguiendo el mayor número de monedas.

Al finalizar un nivel, el *Framework Infinite Mario Bros* genera una lista de las acciones que realizó el jugador así como el tiempo en el que fueron ejecutadas:

- Tiempo en el que inicia el nivel
- Tiempo de inicio y fin de movimiento a la derecha
- Tiempo de inicio y fin de movimiento a la izquierda
- Tiempo de inicio y fin de salto
- Tiempo de inicio y fin de correr a la derecha

- Tiempo de inicio y fin de correr a la izquierda
- Tiempo de inicio y fin de cada cambio de estado del personaje
- Tiempo en que destruye un bloque
- Tiempo en que obtiene una moneda
- Tiempo en que elimina a un enemigo, así como el tipo
- Tiempo en el que pierde una vida
- Estado en que se encontraba el personaje al perder una vida
- Tiempo en el que termina el nivel
- Estado en el que termina el nivel

Es importante mencionar que al ser un ajuste pasivo, es necesario que el jugador termine el nivel, de lo contrario no se podrán obtener datos acerca de su desempeño.

3.4.2 Sistema de control

En esta fase se filtran los datos obtenidos en el paso anterior con el fin de identificar aquellos que permiten mantener al jugador en un estado de *Flujo*, sin embargo, como se menciona en el capítulo 2 es importante tener en cuenta las dificultades existentes para saber si se encuentra en este estado, ya que los datos de entrada son limitados.

Debido a esto, se propone utilizar los datos referentes a obtener puntos, enemigos eliminados, cantidad de saltos, cantidad de bloques destruidos, tiempo que le toma completar el nivel así como la cantidad de vidas que pierde el personaje. Esto es con el fin de tener elementos necesarios para poder diferenciar algunos de los perfiles descritos en el apéndice A: competidor, explorador, coleccionista y ganador.

Estos datos se pueden categorizar en los que aportan información para definir el resultado y las estadísticas de las acciones del jugador.

- **Resultado:** El resultado de manera puntual consiste en si el jugador logró terminar el nivel. Sin embargo, es importante mencionar datos adicionales como los puntos que obtuvo, los enemigos eliminados, los *powerUp's* obtenidos, el tiempo remanente al llegar a la meta así como las muertes que tuvo el personaje durante el juego y la forma en la que sucedieron.
- **Estadísticas del nivel:** Las estadísticas a considerar son el numero de saltos que realiza, la cantidad y forma de eliminar a los enemigos, los bloques destruidos, las monedas obtenidas y los *powerUp's* coleccionados.

Con el fin de facilitar la lectura de los datos recopilados por este sistema, se muestra a continuación una lista con dicha información:

- Referentes al resultado:
  - Completar el nivel
  - Puntaje obtenido
  - Tiempo utilizado para terminar el nivel
  - Vidas perdidas del personaje
  - Causa de cada vida perdida
- Referentes a las estadísticas:
  - Cantidad de saltos realizados
  - Cantidad de enemigos eliminados
  - Forma en la que eliminó a los enemigos
  - Cantidad de bloques destruidos
  - Cantidad de monedas obtenidas
  - Cantidad de *poweUp*'s coleccionados

3.4.3 Sistema de análisis

Tiene la función de analizar los datos filtrados por el sistema de control para decidir los ajustes que deberán llevarse a cabo al crear el siguiente nivel. El dato más relevante para saber si un jugador no ha salido de la zona de *Flujo* es la cantidad de vidas que pierde durante el nivel, si es alta significaría que la dificultad está por encima de sus habilidades, por otra parte, si el tiempo remanente al terminar el nivel es mucho, significa que la dificultad del nivel es baja.

La dificultad de un nivel de un juego de plataformas es, principalmente, la longitud del nivel, el número de enemigos, la cantidad de precipicios y su longitud, así como de la plataforma anterior y posterior a éste[14].

Es importante permitir al jugador elegir diferentes caminos que lo lleven a la meta, ofreciendo en cada uno diferentes retos, con mejores recompensas de acuerdo a la dificultad. Con el fin de evitar la linealidad de los niveles y ofrecer una sensación de control al jugador, se utiliza el esquema de cápsulas, mencionadas anteriormente (figura 3.4), en donde es posible elegir diferentes caminos para llegar a la meta.

Para calificar la dificultad de un nivel se utiliza una medida de rigor[14], que muestra la tolerancia del nivel en términos de la habilidad mostrada por el jugador para completarlo. Debido a ésto, se asigna un valor a cada elemento del juego, negativo si disminuye la dificultad y positivo si la aumenta, en la figura 3.5 se muestran los valores de cada elemento[14].

Elemento	Tipo	Valor
Precipicio	plataforma anterior	-0.25^*
	precipicio	$+0.50^*$
	plataforma posterior	-0.25^*
Enemigos	Goomba	$+0.30$
	Green Koopa	$+0.40$
	Red Koopa	$+0.50$
	Spiny	$+1.00$
	Fly Goomba	$+0.80$
	Fly Green Koopa	$+0.90$
	Fly Red Koopa	$+1.00$
	Piranha Plant	$+0.70$
	Turtle Cannon	$+0.80$
PowerUp	Super Mushroom	-1.00
	Fire Flower	-1.00
Coleccionables	Coin	-0.01

FIGURA 3.5: Valores de rigor para cada elemento del videojuego *Infinite Mario Bros*.
*Los valores se multiplican por el numero de bloques que los componen.

La calificación del rigor del nivel consiste en tomar el promedio del valor de rigor del contenido de cada celda que lo compone, este resultado permite realizar comparaciones entre niveles, ya que la longitud al ser un parámetro que afecta la dificultad puede variar[14].

El análisis de los datos recabados por parte del sistema de control consiste en obtener resultados para ajustar la dificultad de los elementos que contiene cada cápsula, en donde las modificaciones a tener en cuenta son:

- **Longitud del nivel:** si el nivel es demasiado largo, el tiempo remanente será poco, por el contrario, si es mucho, el nivel podría ser muy corto.
- **Cantidad y longitud de los precipicios:** si las muertes del personaje son debido a caídas, es necesario disminuir la dificultad de los precipicios, es decir, que la longitud del precipicio sea menor y mayor la de las plataformas anterior y posterior, así como reducir la cantidad.
- **Cantidad y dificultad de los enemigos:** si las muertes del personaje están dadas por enemigos, es necesario disminuir la cantidad de éstos, así como cambiar el tipo de enemigo a uno con dificultad menor, es decir, disminuir la probabilidad de que aparezca con atributo de volar.

- **Cantidad de *PowerUp's*:** el agregar este tipo de elementos reduce la dificultad del nivel, solo en respuesta a los enemigos, ya que al tomar un elemento de *PowerUp*, el personaje puede resistir hasta dos golpes de los enemigos, sin embargo, esto no aplica para los precipicios.
- **Cantidad de monedas:** estas se colocan en respuesta a las vidas que ha perdido el jugador, para mantenerlo con aproximadamente tres vidas, si tiene menos, aumentan las monedas dentro del nivel, de lo contrario, aparecen con menor probabilidad.

Los tipos de ritmos se encuentran definidos con anterioridad por parte del diseñador, en los cuales el jugador debe realizar una serie de acciones para lograr la experiencia de juego deseada, como se muestra en la figura 3.3. Sin embargo, el ajuste del nivel permite modificar las características de los patrones definidos al cambiar la cantidad y longitud de los precipicios, cantidad y tipo de enemigos, cantidad de *PowerUp's* y la cantidad de monedas dentro del nivel.

Para definir los ritmos se utiliza una gramática libre de contexto, la cual tiene como parámetros los valores de modificación para ajustar la dificultad de cada elemento colocado dentro del nivel[14]. Cabe mencionar que se deben realizar validaciones para garantizar que el nivel sea terminable, principalmente en la longitud de los precipicios, ya que de existir uno de longitud mayor a la máxima distancia que el personaje puede saltar, el nivel es imposible de terminar.

Las cápsulas deben estar colocadas de manera que sea posible para el personaje transitar del final de una al inicio de la siguiente adyacente. Esto se logra con el final del ritmo, en donde el jugador tendrá un momento en donde no se requiere acción alguna para preparar las acciones a realizar en el siguiente ritmo.

Finalmente se debe colocar una plataforma donde no se requiera acción alguna para el inicio del nivel y una meta al final, como parte de los elementos del juego, ya que estas no varían de acuerdo a la dificultad y siempre deben existir.

3.4.4 Sistema del juego

Este sistema recibe los parámetros de la gramática con las modificaciones y se encarga de generar un nivel que cumpla las características requeridas, colocando los elementos en la posición adecuada y así generar un nuevo nivel personalizado para el jugador.

El nivel debe incorporar los elementos formales (procedimientos, reglas, límites y resultados) para que el juego funcione de manera adecuada cuando sea presentado al jugador, así como recabar todos los datos de la sesión de juego para continuar con el ciclo de ajuste pasivo de la dificultad.

Capítulo 4

Una solución procedural para generar niveles en *Infinite Mario Bros*

El objetivo de este capítulo es presentar un algoritmo para implementar un generador de contenido que permita crear niveles del videojuego *Infinite Mario Bros* de manera procedural, el cual pertenece al género de plataformas en dos dimensiones, sin embargo, al cumplir las propiedades requeridas para una solución procedural (confiabilidad, controlabilidad, expresividad y diversidad), es posible aplicar este algoritmo a otros videojuegos del mismo género que tengan elementos formales y mecánica de juego similares.

Utilizando la jerarquía de un nivel del género de plataformas (figura 3.1), el primer paso consiste en crear la estructura de las cápsulas que le darán forma, para esto se utiliza un algoritmo genético, el cual permite adaptar los resultados a un valor esperado, así como mostrar cierto grado de innovación referente a la forma del nivel y su linealidad.

Posteriormente, para crear el contenido de cada cápsula se utiliza una gramática libre de contexto, la cual describe cada elemento del juego, posición, atributos y dificultad, manteniendo las reglas del juego y así, evitar crear niveles que sean imposibles de terminar y con un nivel de dificultad adaptada para cada jugador, creando experiencias de juego personalizadas y únicas.

4.1 Algoritmo genético para generar la estructura de cápsulas

La evolución biológica es el resultado de un método para buscar las mejores soluciones entre una gran cantidad de soluciones parciales. Este conjunto de posibilidades son las secuencias genéticas, y las soluciones deseadas son los elementos con mayor grado de adaptación al entorno en donde se desarrollan y tienen una mayor probabilidad de pasar estas características a su descendencia[10].

El conjunto de reglas que permiten la evolución, desde un punto de vista de alto nivel son sencillas: las especies evolucionan debido a variaciones aleatorias de su código genético

(mutación, recombinación, y otros operadores), posteriormente, una selección natural, en donde los elementos mejor adaptados tienden a sobrevivir y reproducirse, de esta manera se propagan las mejores características a través de la población a futuras generaciones[10].

Algunos sistemas requieren tener la habilidad de adaptarse para funcionar en un entorno que cambia a través del tiempo, otros deben presentar cierto grado de innovación, para construir algo nuevo y original, mientras que otros requieren poder presentar soluciones a problemas que son complejos para ser programadas paso a paso[10].

Al programar reglas sencillas, desde una perspectiva *buttom-up*, las cuales tendrían la función de selección natural, y al incorporar variaciones debido a la cruza o mutación, se espera que un comportamiento emergente ofrezca soluciones de alta calidad a un problema dado, con la capacidad de adaptarse a un entorno variable, utilizando una función de ajuste[10].

Como una solución procedural requiere que el contenido generado cumpla con cierto grado de expresividad y variación, se eligió utilizar algoritmos genéticos para crear la estructura de las cápsulas y así lograr obtener niveles únicos e innovadores, en donde la función de ajuste estará definida por la habilidad del jugador en conjunto con el grado de expresividad del generador e irá adaptándose a los cambios durante el juego.

Los pasos para llevar a cabo un algoritmo genético son:

1. Codificación de los elementos de la población.
2. Definir una población inicial.
3. Calcular la función de ajuste de cada elemento de la población.
4. Repetir hasta que se obtenga una población de tamaño N:
   (a) Seleccionar elementos de la población.
   (b) Cruzar dos elementos con probabilidad P_c.
   (c) Mutar los genes de los nuevos elementos con probabilidad P_m.
5. Reemplazar la población actual por la nueva.
6. Regresar al paso 2.

La condición de término está dada al alcanzar o sobrepasar un cierto valor de ajuste de algún elemento de la población o un número determinado de iteraciones, donde se seleccionará el elemento que tenga mayor ajuste dentro de la población.

4.1.1 Codificación de las cápsulas

El primer paso para poder utilizar un algoritmo genético es codificar la información de los elementos que van a ser parte de la población en estructuras de datos aptas para poder aplicarse operadores de mutación, cruza y evaluar utilizando la función de ajuste. Esta codificación tiene el nombre de *genotipo*[1], mientras que su interpretación es conocida como *fenotipo*[2].

De acuerdo a la jerarquía de elementos que componen a un nivel (figura 3.1), un nivel esta compuesto por cápsulas, utilizando los patrones mostrados en la figura 3.4, cada cápsula se puede ver como un vértice en un gráfica (cada una contiene un ritmo, el cual será generado por la gramática), y las aristas son el inicio y final de cada ritmo contenido en las cápsulas, que son zonas donde no se requiere alguna acción por parte del jugador (ver figura 3.3) y permiten la transición del personaje a cada cápsula adyacente.

Para la codificación del nivel formado por cápsulas, se utilizará una matriz de números binarios, 0 y 1, en donde 0 significa un espacio vacío y 1 representa la existencia de una cápsula. El número de columnas es la cantidad de cápsulas de longitud que posee el nivel, mientras que el número de filas es la altura máxima de la matriz de cápsulas, en este caso será de cuatro, ya que, por la mecánica del juego y que el objetivo es llegar a la meta que se encuentra al final en un tiempo limitado, será suficiente para permitir variaciones y retos dentro del nivel que se presenta al jugador.

$$
\begin{matrix}
0000010000000000000100011100000 \\
0101101000000000011100111111000 \\
0011011001101001111001000001 10 \\
1111100111111011011111111111111
\end{matrix}
$$

FIGURA 4.1: Codificación de las cápsulas que componen un nivel, 0: espacio vacío, 1: cápsula.

El fenotipo de esta matriz es una gráfica bidireccional, en donde los vértices son las casillas que contienen un valor de 1, y las aristas están dadas por las adyacencias de estos valores utilizando la vecindad de *Moore*(figura 4.2), en donde existe un vértice desde la celda i, j si su valor es 1 a cualquier otra celda que tenga un valor de 1 dentro de la vecindad. Es importante que la gráfica generada por el genotipo sea conexa, para que el personaje pueda transitar entre cualquiera dos vértices de la gráfica (definición 1), de lo contrario, habrían elementos inalcanzables y en el peor de los casos el nivel sería imposible de completar.

Definición 1. *Sea G una gráfica, es conexa si, para cualquier par de vértices $u, v \in G$, existe al menos una trayectoria (una sucesión de vértices adyacentes que no repita vértices) de u a v.*

[1] Un conjunto particular de genes
[2] Características físicas de un elemento, como color de ojos, altura e inteligencia

i-1, j+1	**i, j+1**	**i+1, j+1**
i-1, j	**i, j**	**i+1, j**
i-1, j-1	**i, j-1**	**i+1, j-1**

FIGURA 4.2: Se define como vecindad de *Moore* al conjunto de las ocho celdas que rodean una central en una matriz bidimensional.

En la figura 4.3 se puede observar el fenotipo de esta matriz, en donde para cada patrón de cápsulas, existe una configuración de la vecindad de *Moore* con valores binarios.

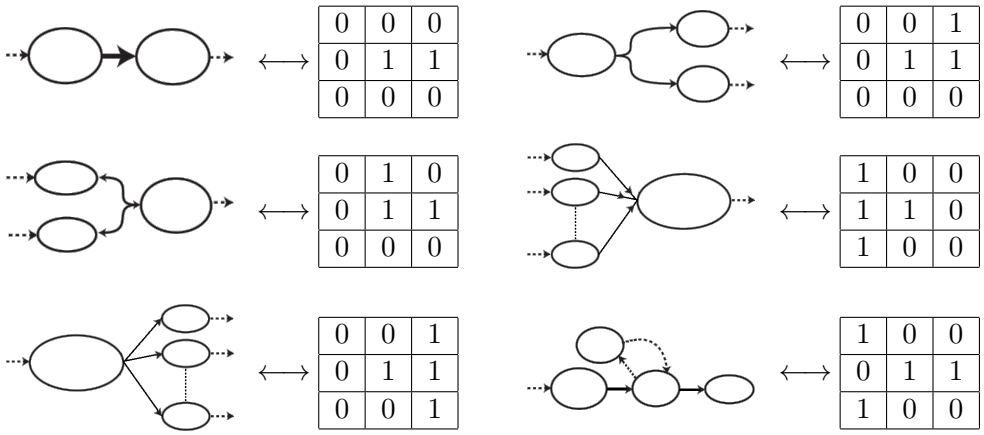


FIGURA 4.3: Fenotipo para las configuraciones de la vecindad de *Moore*.

Dado que el inicio y final del nivel siempre deben existir y no varían al lo largo de todo el juego, se colocan una vez creada la estructura del nivel.

4.1.2 Inicialización de la población

Se genera una población de 100 individuos, en donde cada uno tiene en la última fila de la matriz valores de 1, esto con el fin de garantizar que todos los elementos cumplen con la condición de que el personaje puede transitar de la primer cápsula a la última.

Por lo que la población consiste de tener 100 matrices de tamaño $4 \times N$, donde N puede variar, con valores binarios en cada celda, 1's en la fila 4 y 0's en las otras tres filas.

Esto permite tener individuos cuya interpretación cumple con las características requeridas para que sea terminable el nivel y tienen la estructura más sencilla, con un nulo valor de expresividad.

$$
\begin{array}{c}
00000000000000000000000000000000 \\
00000000000000000000000000000000 \\
00000000000000000000000000000000 \\
11111111111111111111111111111111
\end{array}
$$

FIGURA 4.4: Codificación de los elementos de la población inicial.

4.1.3 Función de ajuste

Para el correcto funcionamiento de un algoritmo genético, es necesario tener una función que mida el desempeño de los individuos de la población enfocada al problema a resolver. Esta función obtiene un resultado numérico a partir de un individuo, lo cual permite realizar comparaciones entre ellos. Las características que evalúa esta función son:

- Gráfica conexa
- Linealidad

La característica más importante a evaluar es si la gráfica representada de cada elemento de la población es conexa, de lo contrario el personaje no podrá transitar desde la cápsula inicial a cualquier otra cápsula del nivel (definición 1), teniendo elementos inalcanzables y en el peor caso sería imposible de terminar. Por lo que la función de ajuste para esta característica es binaria, si la gráfica no es conexa el resultado es 0, de lo contrario, se procede a evaluar la linealidad.

Medir la linealidad de un nivel del videojuego *Infinite Mario Bros* creado por el generador de contenido permite mostrar su capacidad de expresividad y variación[14]. En el caso de un nivel que no es lineal, requiere que el jugador realice más saltos, dando la oportunidad de alcanzar plataformas más altas y presenta más de un posible camino para llegar a la meta.

La linealidad del nivel es medida al realizar una regresión lineal, tomando la posición de cada cápsula dentro de la matriz como valor, es decir número de columna y fila, posteriormente se califica el nivel tomando la suma de los valores absolutos de la distancia entre el número de fila de la cápsula y su valor esperado a la línea, dividido por el número total de cápsulas. Los resultados son normalizados al intervalo $[0, 1]$, en donde 0 es totalmente lineal y 1 es altamente no lineal (figura 4.5). En *Infinite Mario Bros*, las colinas de diferentes alturas pueden ser colocadas una encima de otra, permitiendo al personaje alcanzar lugares más altos y agregando nuevos patrones al diseño del nivel.

La función de ajuste evaluará a los elementos de la población y aquellos que estén más cercanos al valor de linealidad definido por el sistema de análisis, tendrán una mejor calificación.

$$
\begin{array}{l}
11111111111111111111111111111111 \\
11111111111111111111111111111111 \\
11111111111111111111111111111111 \\
11111111111111111111111111111111
\end{array}
$$

$$
\begin{array}{l}
11111111111111111111101111011 \\
10111110111111111110111111111 \\
01011111111111101111111111011 \\
10100010111000110001011000011 0
\end{array}
$$

$$
\begin{array}{l}
00000000000000000000000000000000 \\
00000000000000000000000000000000 \\
00000000000000000000000000000000 \\
11111111111111111111111111111111
\end{array}
$$

FIGURA 4.5: Tres ejemplos de estructuras de niveles con diferentes valores de linealidad, 1 en el primero, 0.99 en el segundo y el tercero con 0.

4.1.4 Selección

El algoritmo de selección tiene la función de elegir a los individuos que podrán combinar sus características para generar a los elementos de la nueva generación. El propósito de esta función es enfatizar el ajuste de los individuos de la población para que la nueva generación disponga de elementos con un mayor nivel de adaptación.

La selección debe tener un balance entre mutación y cruza, exploración y explotación respectivamente. Una selección muy demandante llevará a que los elementos sub-óptimos con el mayor ajuste dentro de la población predominen, eliminando la diversidad para progresar. Por otro lado, una selección muy débil resultará en una evolución lenta[10].

Se utilizará un algoritmo de selección por torneo, el cual consiste en elegir dos individuos al azar dentro de la población, posteriormente se genera un número aleatorio $r \in [0, 1]$. Si $r < k$ (donde k es un parámetro fijo, generalmente 0.75), el individuo con un mayor grado de ajuste es seleccionado para ser un padre; de lo contrario el que tenga menor valor será elegido. Los dos elementos son regresados a la población original y pueden volver a ser seleccionados[10].

Una variación a este algoritmo consiste en seleccionar un número arbitrario de elementos de la población p, donde $p \geq 2$. Al variar el número de individuos que participan en cada torneo se puede modificar la presión de selección. Cuando participan muchos individuos en cada torneo, la presión es alta y los peores tienen una probabilidad baja de ser elegidos.

La complejidad del proceso de selección es $O(n)$, para el caso de $p = 2$, ya que se eligen al azar dos elementos y se realiza una comparación entre ellos. Por otro lado si $p = N$, donde N es el tamaño de la población, se tendría una complejidad de $O(n^2)$, ya que para elegir el elemento mínimo o máximo de N tiene una complejidad de $O(n)$.

Adicionalmente a la selección de los individuos para la fase de cruza, se copian a la nueva población los mejores $k = 10\%$ de la población, esto con el fin de evitar que se pierdan debido a la fase de cruza o que no sean elegidos para reproducirse. Esto tiene el nombre de Elitismo, y ha mostrado mejorar el desempeño de los algoritmos genéticos de manera significativa [10].

4.1.5 Cruza

La finalidad de cruzar dos individuos de la población es obtener un nuevo elemento que contiene características de ambos, esperando que aumente su valor de ajuste.

Dada la codificación de los elementos de la población, y el requerimiento de que la gráfica generada por el genotipo sea conexa, los puntos de corte para combinar los genes de dos elementos, serán cortes verticales, con esto se puede asegurar que cada bloque cumple con ser conexo, y cuya cantidad puede ir de 1 a $N - 1$ puntos. Cada bloque de la matriz puede ser elegido para generar al nuevo elemento con la misma probabilidad $P_c = 0.7$.

Una vez realizada con éxito la cruza de dos individuos, es importante verificar que la gráfica generada sea conexa, en caso de no ser así, con el fin de no descartar directamente resultados por esta razón, se debe verificar, en la última columna de p_i y en la primer columna de p_{i+1} que al menos un 1 de p_i sea adyacente a un 1 de p_{i+1}.

En caso de no poder transitar de p_i a p_{i+1}, es necesario agregar valores de 1 en la primer columna de p_{i+1} hasta lograr que sea adyacente con al menos un 1 de la última columna de p_i.

Por otra parte, todos los $1's$ de cada p_i cumplen con ser adyacentes a otro 1.

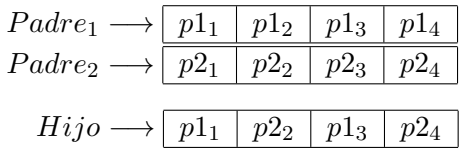


FIGURA 4.6: Nuevo elemento, formado la combinación de bloques de dos elementos.

4.1.6 Mutación

La mutación permite variar algunos de los genes de un individuo de manera aleatoria. Aunque es posible elegir elementos de la población actual directamente y mutarlos antes de introducirlos a la nueva población, la mutación se utiliza generalmente en conjunto con la fase de cruza, en donde, si es generado un nuevo individuo, el descendiente es en donde se aplican los operadores de mutación con probabilidad P_m, cuyo valor usualmente es cercano al 1%.

El valor de la probabilidad de mutación es bajo, ya que lo más común es que al realizar este cambio, el individuo disminuya su capacidad de ajuste, sin embargo, el aplicar la función de mutación permite que todo el espacio de búsqueda tenga posibilidad de ser examinado.

Es importante mantener el balance entre los operadores de cruza, mutación y selección para lograr un correcto funcionamiento del algoritmo genético. Esto depende de una correcta codificación de los individuos y los detalles que verifica la función de ajuste[10].

El valor de cada celda puede mutar con probabilidad $P_m = 0.1$, sin embargo, es necesario tener en cuenta que el aplicar este operador se evite generar un bloque que no cumpla la regla de tener una gráfica conexa y pueda llevar a tener bloques inalcanzables o que sea imposible terminar el nivel. Por lo que el mutar de $0 \rightarrow 1$ no genera problemas siempre que exista un 1 adyacente utilizando la vecindad de *Moore*, pero en el caso de cambiar de $1 \rightarrow 0$, solo podrá suceder cuando sea posible transitar de la primera fila a la tercera, en la vecindad de *Moore*, después de la modificación.

4.1.7 Reemplazo

La nueva población estará compuesta por los elementos resultantes de los operadores de cruza y mutación cuando se cumple la probabilidad de de cruza P_c, de lo contrario pasan los padres directamente.

También se incluyen los elementos elite que fueron elegidos durante la fase de selección, esto con el fin de asegurar que los mejores individuos no se pierdan al no ser elegidos y pasen a la nueva generación.

4.1.8 Condición de término

Para concluir el proceso del algoritmo genético se debe alcanzar o sobrepasar el valor definido por el sistema de análisis referente a la linealidad del nivel buscada, con una tolerancia de 0.0001; en caso de no llegar a este valor, se detendrá el programa una vez alcanzados los 100 ciclos y se tomará el elemento con el mayor valor de ajuste.

Es importante tener en cuenta el tiempo que se genera un nuevo nivel, el cual debe ser pequeño, ya que de lo contrario se corta la sensación de fluidez y el jugador podría dejar de jugar.

4.2 Gramática para la generación del contenido de las cápsulas

La representación de los niveles en *Infinite Mario Bros* es un arreglo bidimensional de todos los objetos del juego definidos en los elementos formales, como enemigos, plataformas, tuberías,

cañones, cajas y monedas. Donde cada uno tiene propiedades, así como una representación visual que lo diferencia de los otros, para facilitar saber cuál es su función, y coordenadas para definir su posición.

Para poder permitir variaciones en el diseño del nivel, se utiliza una gramática libre de contexto, con el fin de generar un conjunto de bloques dentro de cada cápsula de la estructura obtenida por el algoritmo genético mencionado anteriormente. Cada cápsula contiene un vector de longitud variable, el cual es creado utilizando las reglas de producción de la gramática, lo que genera una representación sintácticamente correcta dentro del dominio del problema[14].

Para ayudar a mantener al jugador dentro de un estado de *Flujo* se utilizan ritmos (capítulo 3), los cuales son una agrupación de bloques definidos por el diseñador del juego, en donde se espera que el jugador realice ciertas acciones. Los parámetros de los bloques dentro de la gramática permiten modificar los tipos y dificultad de éstos.

La dificultad esperada es definida por el sistema de análisis, en donde se califica el rigor del nivel con base en el desempeño del jugador, y el nuevo nivel es creado con elementos cercanos a dicho valor, con el fin de tener un nivel con un promedio de dificultad cercano a éste.

Como parte de los requisitos de una solución procedural, se deben tener en cuenta las restricciones dentro de la gramática para no generar obstáculos imposibles de superar, como precipicios o tuberías que eviten que el personaje pueda llegar a la meta, así como un corto tiempo de respuesta para ofrecer al jugador el siguiente nivel con un valor de expresividad que ofrezca diferentes retos.

4.2.1 Gramática

La función de la gramática es crear los elementos del nivel al colocar bloques con al menos dos propiedades (x, y), números que especifican las coordenadas de un bloque en un arreglo bidimensional. Una regla importante a tener en cuenta es que todos los elementos deben ser alcanzables y que el personaje pueda transitar desde el inicio del nivel hasta la meta utilizando las acciones definidas en los elementos formales del juego.

De acuerdo a los diseñadores de juegos, la longitud de las plataformas anterior y posterior tienen un rol importante en la experiencia de juego. La longitud de una plataforma anterior a un precipicio afecta la dificultad del juego, ya que algunas veces es necesario utilizar en conjunto las acciones de correr y brincar para obtener un salto de mayor distancia y superar precipicios amplios[14]. Para esto se agregan dos parámetros especificando la longitud de la plataforma anterior y posterior a cada precipicio.

Para obtener todos los elementos definidos por elementos formales, es necesario tener las producciones de la gramática que los generen, así como los parámetros adecuados para poder

colocarse. Las plataformas y los bloques solo requieren las coordenadas y el tipo, mientras que otros elementos requieren adicionales como la altura en el caso de las tuberías, cañones y colinas.

Para lograr ésto, es necesario definir una gramática libre del contexto que permita definir los elementos de las posibles soluciones de un nivel de *Infinite Mario Bros* y asignar cadenas de números a soluciones sintácticamente correctas. Cada producción recibe parámetros, los cuales deben estar restringidos para no generar obstáculos que sean imposibles de superar.

Definición 2. ***Gramática libre del contexto o tipo 2***
Una gramática $G = N, T, P, S$ *es una gramática libre del contexto o tipo 2, donde:*

$N ::= Conjunto\ finito\ de\ simbolos\ no\ terminales.$
$T ::= Conjuntofinito\ de\ simbolos\ terminales.\ N \cap T = \varnothing$
$P ::= Conjunto\ finito\ de\ producciones.$
$S ::= Simbolo\ inicial.$

Y sus producciones son de alguna de las siguientes formas:

$S ::= \varepsilon$
$A ::= \alpha$

con $S, A \in N$*,* S *símbolo inicial y* $|A| \leq |\alpha|$*,* $\alpha \in (N \cup T)^+$ *y si la producción* $S ::= \varepsilon$ *está en la gramática,* S *no aparece del lado derecho de ninguna producción.*

En la figura 4.7 se describe la gramática propuesta para construir una solución y definir todos los elementos de los niveles de *Infinite Mario Bros*. Es importante notar que las producciones no necesariamente están ordenadas con respecto a los valores de las posiciones ya sea en x o y.

Se incluyen los valores de w_{before} y w_{after}, que son utilizados para definir las longitudes de las plataformas anterior y posterior respectivamente y w para la longitud del vacío. Mientras que en otros elementos se define el parámetro h que sirve para marcar la altura que tendrá dicho elemento. Un ejemplo generado por esta gramática es: $cannon(10, 4, 3)platform(15, 3, 4, 1, grass)tube(62, 4, 3)$, con lo cual se obtiene una plataforma de tipo pasto, un cañón de altura 4 y una tubería de altura 3.

Uno de los conflictos resultantes es que cada bloque tiene asignadas la coordenadas de x, y, lo cual puede llevar a que existan traslapes en algunos elementos del nivel. Para esto se asigna un valor de prioridad a cada tipo de elemento, y cuando existe traslape entre dos elementos se mantiene el de mayor prioridad.

Para colocar a los enemigos es necesario primero construir la estructura física del nivel, y posteriormente calcular las posibles posiciones en donde se puede colocar un enemigo. Cada

$level \ ::= \ \langle cell \rangle$

$\langle cell \rangle \ ::= \ \langle rhytm \rangle \ | \ \langle rhytm \rangle \langle cell \rangle$

$\langle rhytm \rangle \ ::= \ gap(\langle x \rangle, \langle y \rangle, \langle w \rangle, \langle w_{before} \rangle, \langle w_{after} \rangle, \langle platform_{type} \rangle)$
$| \ platform(\langle x \rangle, \langle y \rangle, \langle w \rangle, \langle h \rangle, \langle platform_{type} \rangle)$
$| \ hill(\langle x \rangle, \langle y \rangle, \langle w \rangle, \langle h \rangle)$
$| \ block(\langle x \rangle, \langle y \rangle, \langle w \rangle, \langle block_{type} \rangle)$
$| \ cannon(\langle x \rangle, \langle y \rangle, \langle h \rangle)$
$| \ tube(\langle x \rangle, \langle y \rangle, \langle h \rangle)$
$| \ enemy(\langle x \rangle, \langle y \rangle, \langle enemy_{type} \rangle)$
$| \ coin(\langle x \rangle, \langle y \rangle, \langle w \rangle, \langle h \rangle)$

$\langle platform_{type} \rangle \ ::= \ grass \ | \ rock \ | \ bricks \ | \ stone \ | \ metal \ | \ wood$

$\langle enemy_{type} \rangle \ ::= \ goomba \ | \ greenKoopa \ | \ redKoopa \ | \ spiny \ | \ flyGoomba \ | \ flyGreenKoopa \ | \ flyRedKoopa$

$\langle block_{type} \rangle \ ::= \ block(\langle content \rangle) \ | \ misteryBlock(\langle content \rangle)$
$\langle content \rangle \ ::= \ powerUp \ | \ coin \ | \ empty$

$\langle w_{after} \rangle \ ::= \ \langle x \rangle$
$\langle w_{before} \rangle \ ::= \ \langle x \rangle$
$\langle w \rangle \ ::= \ \langle x \rangle$
$\langle h \rangle \ ::= \ \langle y \rangle$
$\langle x \rangle \ ::= \ [1 \ldots n]$
$\langle y \rangle \ ::= \ [1 \ldots 15]$

FIGURA 4.7: Gramática para generar todos los elementos de un nivel.

uno de éstos forma parte de la gramática y utiliza los parámetros de posición x, y así como el tipo de enemigo que será y un atributo para definir si tendrá alas o no.

4.2.2 Ritmos

Los ritmos son fundamentales para crear una fluida experiencia de juego, además de contribuir de manera considerable en la dificultad de los videojuegos de plataformas con retos basados en precisión[17]. Los ritmos son pequeños conjuntos de elementos que generan un reto, el cual consiste en ciertas acciones que debe realizar el jugador y la gramática que lo define. Las monedas, además de aumentar los puntos del jugador, pueden ser utilizadas para guiar las acciones esperadas del jugador.

Cada cápsula de la estructura del nivel tiene una longitud fija de 15 unidades, en donde se colocan elementos del nivel para generar el ritmo. Dependiendo de la linealidad de cada nivel, el jugador podrá elegir entre diferentes caminos para llegar a la meta, y cada cápsula deberá ofrecer diferentes retos y recompensas.

Cada ritmo posee dos propiedades: tipo y densidad. Para generar un ritmo primero se asigna un valor a cada una de estas propiedades basadas en los datos proporcionados por el sistema de análisis.

El material del cual están compuestas las plataformas puede ser de diferentes elementos como bloques sólidos donde el personaje puede caminar, existen otros elementos en donde también es posible transitar y tienen características adicionales, como colinas, en donde el personaje puede utilizarlas para subir a un nivel mas alto, o bloques destructibles, los cuales pueden tener monedas o *powerUp's*. Otro tipo son los cañones, los cuales pueden disparar balas, que son un tipo de enemigo, y tuberías que pueden contener una planta piraña que es otro tipo de enemigo. El colocar enemigos con cierta distancia entre cada uno, exige al jugador realizar cierta cantidad de saltos, ya sea para eliminarlos o evadirlos, por lo que también es considerado como un ritmo.

Las monedas, además de contar como puntos para el jugador, se pueden utilizar para guiar las acciones que se espera que realice en cada ritmo (ver figura 3.3).

Ritmo	Tipo	**Regular**: saltos a distancias iguales. **Oscilatorio**: alterna saltos cortos seguidos de largos. **Aleatorio**: saltos a distancias aleatorias.
	Densidad	Valores en el rango $[0, 1]$, baja a alta, respectivamente.
Composición	Plataforma	Elemento básico para transitar.
	Colina	Permiten al personaje subir a plataformas mas altas.
	Bloques	Son destructibles o pueden tener algún objeto.
	Cañón	Permite al personaje pararse sobre uno y es un tipo de enemigo.
	Tubería	Permite al personaje pararse sobre uno y puede contener un enemigo.
	Enemigos	El personaje debe brincar sobre éstos para eliminarlos o evadirlos.
Monedas	Recompensa	Monedas sobre un plataforma.
	Ritmo	Cantidad de monedas sobre un precipicio para marcar el ritmo de los saltos.

FIGURA 4.8: Parámetros para la generación de los ritmos.

Para definir los ritmos, es importante tener en cuenta las capacidades de salto del personaje, ya que el crear vacíos de longitudes muy grandes puede evitar que el jugador pueda superarlos y así tener un nivel imposible de terminar. La dificultad de los ritmos depende del tipo, uno regular será más sencillo de llevar que uno totalmente aleatorio.

Como se mencionó en el capítulo 3, la transición entre cada grupo de ritmos, cada uno contenido en una cápsula, debe tener un momento de pausa, el cual delimita el ritmo y permite al jugador pensar en su siguiente grupo de acciones, por lo que se debe agregar una plataforma en la que no se deba realizar acción alguna para evitar daños al personaje y a su vez, permita la transición del personaje, sin embargo, sí es posible tener objetos como bloques con recompensa o monedas.

4.2.3 Ajuste de dificultad

Con el fin de ajustar la dificultad del nivel se utiliza en conjunto el algoritmo genético, que se encarga de crear la estructura de las cápsulas; la gramática, que define todos los elementos del nivel; y los ritmos, donde cada uno tiene parámetros para modificar utilizando los resultados del sistema de análisis.

Como se menciona en el capítulo 3, el sistema de análisis genera resultados para modificar el contenido del nivel referente a la longitud del nivel, características de los precipicios, enemigos, *PowerUp's* y monedas.

A continuación se describe de manera detallada cómo se califica cada una de las características que se deben modificar por parte del sistema de análisis. Cada calificación puede tener impacto en la misma celda y generar cambios en diferentes elementos que la componen, ya sea en el tipo de ritmo, en las recompensas que ofrece y en el tipo de enemigos que contiene.

Longitud del nivel

El tiempo total para completar un nivel es de 200 segundos, de lo cual se espera que el jugador tarde aproximadamente 100 segundos en llegar a la meta con una tolerancia de ± 10 segundos. Por lo que si tarda menos de 90 segundos el nivel debe ser más largo, por el contrario, si tarda más de 110 segundos, se deberá reducir la longitud de éste.

Cada cápsula tiene una longitud de 15 bloques, por lo que para obtener el número de columnas a generar por el algoritmo genético, se debe dividir la longitud entre esta cantidad.

$$N_{columns} = \begin{cases} \frac{len_{before}}{time_{before}} \times 100 \times \frac{1}{15} & si \quad time < 90 \ \vee \ 110 < time \\ len_{before} \times \frac{1}{15} & si \quad 90 \leq time \leq 110 \end{cases}$$

FIGURA 4.9: Longitud del nuevo nivel. len_{before}: longitud del nivel anterior. $time_{before}$: tiempo utilizado para completar el nivel.

Se obtiene la velocidad promedio del personaje para completar el nivel y se multiplica por el tiempo esperado para terminar cada nivel.

Linealidad

Además de la cantidad de columnas de las cápsulas, el algoritmo genético requiere un valor para ajustar la linealidad de la estructura, este valor definirá la cantidad de caminos que puede haber para llegar a la meta, tiene valores entre 0 y 1, donde 0 es totalmente lineal y 1 es no lineal, el cual aumentará si el jugador realiza una gran cantidad de saltos con respecto a los 5 niveles anteriormente jugados y disminuye si realiza un número de saltos menor

$$linearity = \begin{cases} linearity_{before} + 0.02 & si \quad avgJumps \leq jumps \\ linearity_{before} - 0.02 & si \quad avgJumps > jumps \end{cases}$$

$$avgJumps = \frac{(avgJumps \times 4) + jumps}{4 + 1}$$

FIGURA 4.10: Valor de linealidad esperado de la estructura de cápsulas, tal que $linearity \in [0, 1]$.

Dificultad de los ritmos

La cantidad de vidas que se otorgan al jugador al inicio del juego es de tres, cuando se pierden algunas a causa de precipicios es necesario reducir la dificultad de éstos, lo cual se logra reduciendo la cantidad y dando prioridad a ritmos de tipo regular.

Los parámetros para modificar la dificultad en los elementos de precipicios, enemigos, *PowerUp's* y monedas, se generan en conjunto, ya que varios elementos impactan en la misma cápsula.

El tipo y densidad del ritmo están dados por la cantidad de vidas perdidas al caer por un precipicio, por lo que es necesario utilizar más ritmos de tipo regular que oscilatorios y aleatorios, además de disminuir su cantidad.

$$density = \begin{cases} density_{before} - (5 \times 0.001 \times lives_{lost}) & si \quad lives_{lost} > 0 \\ density_{before} + 0.001 & si \quad lives_{lost} = 0 \end{cases}$$

$$type = \begin{cases} P(reg) = 0.6, P(osc) = 0.25, P(ran) = 0.15 & si \quad lives_{lost} > 0 \\ P(reg) = 0.4, P(osc) = 0.3, P(ran) = 0.3 & si \quad lives_{lost} = 0 \end{cases}$$

FIGURA 4.11: Ajuste de valores de densidad y tipo de ritmo.

Además de la densidad y el tipo de ritmo, la dificultad es afectada por las características que componen a los elementos de los ritmos, colocar plataformas de tipo colina, metal, madera o roca no modifica la dificultad del nivel, sin embargo, agregar tuberías y cañones como parte de la plataforma aumenta la dificultad, ya que también pueden contener o generar enemigos por donde transita el personaje.

Enemigos

Los ritmos también pueden estar dados por enemigos, donde el personaje debe seguirlo al brincar sobre estos, sin embargo, a diferencia de los precipicios, cuando el personaje es tocado por uno, no necesariamente pierde una vida.

Los parámetros a modificar en este caso son el tipo de enemigo y el atributo de volar, el cual, al estar activado, hace que los enemigos tengan un movimiento más complejo. La forma en que se

agregan al nivel, es eliminando un bloque de precipicio y agregando un enemigo, los valores para esto dependen de la cantidad de vidas que ha perdido el personaje por enemigos y que tipo fue el que lo causó, disminuyendo la cantidad de enemigos de dicho tipo y aumentando la posibilidad de que aparezcan enemigos de los otros tipos.

$$d_{enemy} = \begin{cases} d_{enemyBefore} - (5 \times 0.001 \times death_{enemy}) & si \quad death_{enemy} > 0 \\ d_{enemyBefore} + 0.001 & si \quad death_{enemy} = 0 \end{cases}$$

FIGURA 4.12: Ajuste de valores de densidad y tipo de enemigos, tal que $enemy \in [0, 1]$.

PowerUp

Para compensar la dificultad agregada por los enemigos, la composición de algunas plataformas se modifica para incluir bloques que contengan *powerUp's*, de modo que el personaje al tomarlos cambie de estado a *Super Mario* o *Fire Mario*, permitiéndole resistir hasta dos golpes de los enemigos antes de perder una vida.

Los valores modificados por el sistema de análisis, referentes a estos elementos, consiste en aumentar la cantidad de bloques de *powerUp* si el personaje perdió una o más vidas durante el nivel anterior por causa de un enemigo, de lo contrario se disminuye esta cantidad, sin llegar a cero.

$$d_{powerUp} = \begin{cases} d_{enemyBefore} + (5 \times 0.05 \times death_{enemy}) & si \quad death_{enemy} > 0 \\ d_{enemyBefore} - 0.05 & si \quad death_{enemy} = 0 \end{cases}$$

FIGURA 4.13: Ajuste de valores de densidad de *PowerUp's*, tal que $d_{powerUp} \in (0, 1]$.

Monedas

Las monedas tienen la función de ser una recompensa, aumentan la cantidad de puntos que consigue el jugador, sin embargo, también pueden funcionar para marcar los saltos que se espera que realice durante un ritmo y facilitar avanzar a través de ellos.

Dado que al perder todas las vidas se acaba el juego, y como forma de compensar las vidas perdidas durante los niveles, al juntar 100 monedas el jugador recibe una vida extra.

Por lo que el sistema debe ajustar la cantidad de monedas que aparecen dentro de cada nivel, para que, cuando el jugador tenga pocas vidas le sea más sencillo reponerlas.

$$d_{coin} = \begin{cases} 1 - \dfrac{1}{5 - lives} & si \quad lives < 5 \\ d_{coinBefore} - 0.05 & si \quad lives \geq 5 \end{cases}$$

FIGURA 4.14: Ajuste de valores de densidad de aparición de monedas. $d_{coin} \in [0, 1]$.

Rigor del nivel

Para calificar el nivel, se utilizan los valores de la tabla 3.5, en donde se toma el promedio de los valores de rigor de cada elemento contenido en el nivel. Lo cual, en conjunto con la linealidad y longitud del nivel, permite generar una gráfica del comportamiento de los niveles generados proceduralmente y la habilidad del jugador durante una sesión de juego.

Capítulo 5

Pruebas y resultados

El propósito de este capítulo es mostrar los resultados obtenidos por el generador procedural de niveles para el videojuego *Infinite Mario Bros* descrito en el capítulo 4, a partir de pruebas de usuario, en donde se tomarán los valores de dificultad, linealidad y tiempos de ejecución para observar el comportamiento del programa a lo largo de una sesión de juego.

Primero se muestra la forma en la que el programa arroja los resultados, en donde por cada nivel jugado se genera un archivo que contiene las datos requeridos para poder realizar las observaciones.

Posteriormente se muestra lo referente al diseño de la prueba de usuario, en donde además de requerir jugar, es necesario contestar una encuesta para poder calificar de manera subjetiva la calidad y dificultad del contenido generado.

En la última sección se describe cómo se utilizan e interpretan los datos obtenidos de los jugadores, además de mostrar las propiedades que se cumplen de una solución de generación procedural de contenido.

Es importante tener en cuenta los problemas que pueden existir referentes al ajuste de dificultad utilizando un sistema de ajuste pasivo, como datos indirectos, análisis basados en suposiciones y cambios basados en un sistema rígido (capítulo 2).

5.1 Sesión de juego

Una sesión de juego consiste en los niveles jugados por un jugador, desde el inicial hasta que ha perdido todas sus vidas o decide dejar de jugar. Es importante tener en cuenta que, dado que el sistema ajusta la dificultad del contenido a las habilidades del jugador, se espera que la cantidad de vidas que posee no llegue a cero, por lo que la decisión de abandonar el juego, además de ser por aburrimiento o frustración en caso de que el sistema no funcione de manera adecuada, puede ser porque no quiera dedicar una gran cantidad de tiempo a permanecer jugando.

Cada nivel jugado genera datos que nos permitirán observar el comportamiento de los resultados del videojuego referentes al contenido generado, por lo que es necesario definir el formato en el que se muestran para posteriormente utilizarlos en los resultados.

5.1.1 Datos por nivel

El programa genera un archivo de salida para cada nivel concluido, en texto plano, donde se pueden observar datos referentes al nivel:

- **Estructura de las cápsulas:** Es una matriz de 0's y 1's en donde se muestran las cápsulas y sus adyacencias.
- **Gramática del nivel:** Son las producciones de la gramática descrita en el capítulo 4, con todos sus parámetros en texto plano y permite generar la matriz del nivel.
- **Matriz del nivel:** Está compuesta de caracteres que componen el nivel, y es posible apreciar como es visualmente.
- **Número de nivel:** De manera secuencial, muestra el número de nivel que ha sido jugado durante la sesión.
- **Rigor:** Es la forma de calificar la dificultad del nivel de acuerdo a su contenido, como se menciona en el capítulo 3.
- **Linealidad:** Es el valor que permite medir la capacidad de expresividad y variación del generador de contenido, permitiendo al jugador alcanzar plataformas más altas y teniendo diferentes rutas para llegar a la meta (capítulo 4).
- **Tiempo de generación de contenido:** Es el tiempo que tarda en generar un nuevo nivel, el cual abarca desde que el jugador termina un nivel hasta que le es presentado el siguiente.
- **Muertes por enemigos:** Cantidad de vidas perdidas durante el nivel jugado por enemigos.
- **Muertes por precipicio:** Cantidad de vidas perdidas durante el nivel jugado por caídas en precipicios.
- **Longitud del nivel:** Distancia desde el inicio a la meta, cuya unidad de medida son los bloques. Este valor coincide con el número de columnas de la matriz del nivel.

La figura 5.1 es un ejemplo de los datos que arroja el sistema, cabe mencionar que se generarán tantos archivos como niveles sean concluidos durante una sesión de juego y contiene los datos descritos anteriormente.

Este archivo adicionalmente tiene la función de guardar el nivel generado, ya que tanto la gramática como la matriz del nivel tienen los elementos necesarios para crear el nivel, por lo que de ser necesario, es posible cargase en el sistema y volver a ser jugado.

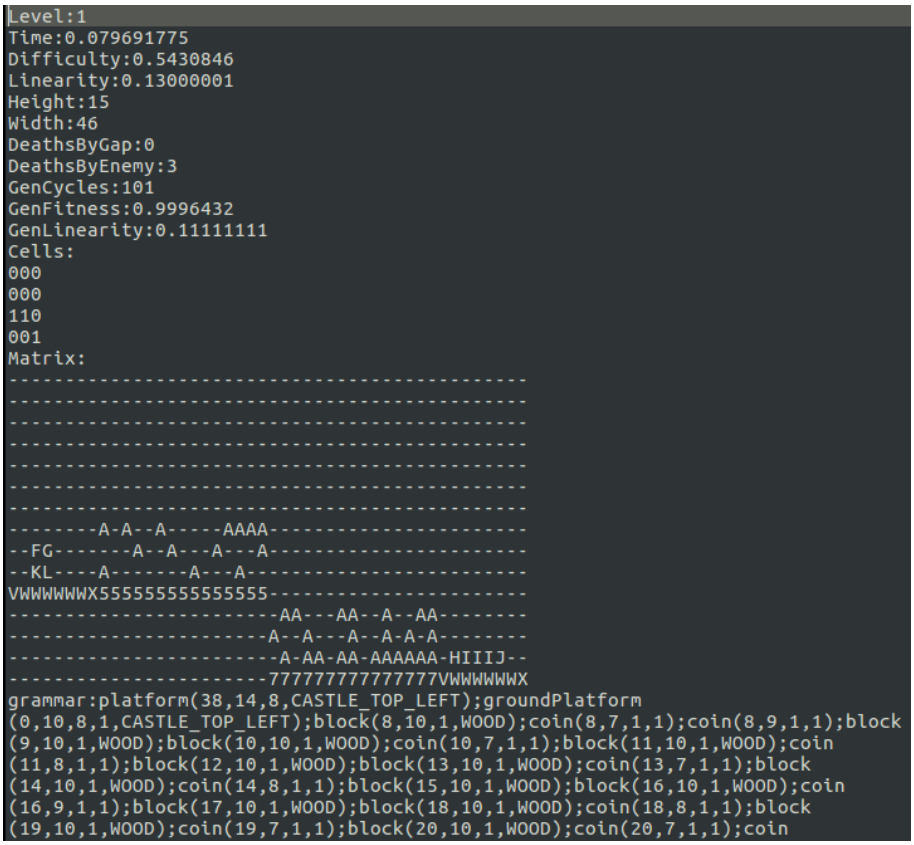


FIGURA 5.1: Ejemplo de datos arrojados por el sistema por cada nivel jugado.

5.2 Pruebas de juego

Como parte del diseño de videojuegos centrado en el juego, realizar pruebas de juego es una parte fundamental para poder lograr obtener los objetivos del producto y tener en cuenta la experiencia de juego del jugador, ya que, como desarrollador, la opinión puede estar sesgada dado el conocimiento que adquiere del juego.

Existen varias formas para realizar este tipo de pruebas, desde informales y cualitativas, a más estructuradas y cuantitativas, las cuales dependen en gran medida de los recursos disponibles para realizarse, ya sea tiempo, dinero o infraestructura.

Para videojuegos como *Halo 3*, *Microsoft Game Studios* realizó pruebas de juego abarcando más de 3,000 horas y con 600 jugadores en una instalación especializada para éste tipo de actividades. Sin embargo, para videojuegos desarrollados por estudios independientes o empresas pequeñas, basta con tener de 10 a 20 jugadores para realizar las pruebas[4].

La importancia de realizar estas pruebas es poder tener retroalimentación por parte de los jugadores de manear objetiva, y así no perder de vista el objetivo y poder mejorar la experiencia de juego en cada iteración. Al ser desarrollado utilizando un proceso iterativo, se espera que los

cambios sean cada vez menores, sin embargo, hay que tener en cuenta que no es posible cambiar el diseño básico de juego en todas las ocasiones, ya que la meta es terminar el producto eventualmente.

De no realizarse esta fase, es posible que no se puedan observar errores importantes de diseño durante el desarrollo, lo cual implicaría que la experiencia de juego no se pueda obtener de la manera esperada, y los cambios que deban realizarse sean difíciles de llevar a cabo en un producto ya terminado, debido a la gran cantidad de líneas de los programas o si afecta una parte del núcleo del juego.

5.2.1 Enfoque de la prueba

Para diseñar una prueba de juego, es necesario enfocarse en qué es lo que se desea mejorar o medir, dado que estas pruebas abarcan diferentes áreas. En el área de mercadeo se utilizan para determinar la población que comprará el juego y estimar cuántas unidades podrán ser vendidas. El área de calidad (*QA*) busca encontrar posibles errores de programación (*bugs*) o de compatibilidad. Los diseñadores de interfaces buscan mejorar la usabilidad y facilitar la interacción. Sin embrago como diseñador, se busca asegurarse que el juego funciona de la manera esperada, es completo internamente y esta balanceado, para lograr la experiencia de juego deseada.

Con esto en mente, el objetivo de esta prueba de juego, es poder observar si los jugadores logran llegar a un estado de *Flujo*, al jugar niveles con ajuste de dificultad basados en sus habilidades, es decir, que no sea muy fácil o difícil y los lleve a experimentar aburrimiento o frustración durante la sesión de juego.

Otro punto importante a tener en cuenta durante la prueba es asegurar que todos los niveles jugados fueron terminables, dado que es posible que lleguen a presentarse errores con baja probabilidad (capítulo 1), y haya diferencias considerables entre cada uno. Ésto con el fin de cumplir de manera subjetiva las cualidades requeridas por una solución procedural (capítulo 3): velocidad, confiabilidad, expresividad y controlabilidad.

5.2.2 Diseño de la prueba

Para la prueba, es necesario solo mencionar lo necesario a los jugadores, como es cómo funciona el juego con una explicación mínima, y permitirles jugar, para poder obtener una perspectiva objetiva.

Durante la sesión de juego, para obtener una retroalimentación objetiva, es necesario observar a la persona desde otro cuarto, ya sea a través de un vidrio con vista de un solo lado o una cámara

de vídeo, en caso de no contar con estas opciones también es posible observar desde atrás, interactuando con ella en la menor cantidad posible.

Para ayudar a tener en mente que durante la prueba el rol debe ser solo de observador, se recomienda tener un guión preparado que incluya lo que se va a decir a lo largo de ésta[4], el cual se describe a continuación:

- **Introducción** $(\leq\ 3\ min)$**:** Dar bienvenida a los jugadores y agradecer por participar. Mencionar de manera breve en qué se está trabajando y cuál es el objetivo. Posteriormente explicar el proceso de prueba de juego y cómo ayuda a mejorar y evaluar el juego. Explicar que la información obtenida es únicamente para referencia y no será mostrada a nadie más.
- **Preguntas de contexto** $(\leq 5\ min)$**:** Presentar la primera parte del cuestionario, en donde se busca definir qué tipo de jugador es, su experiencia en videojuegos del género de plataformas, cuáles son sus videojuegos preferidos y la cantidad de tiempo que dedican a jugar.

  Antes de pasar a la siguiente fase, mostrar los controles para poder jugar (figura 5.2), las acciones que puede realizar el personaje y, de ser necesario, el objetivo del juego.
- **Sesión de juego** $(\leq 15\ min)$**:** Explicar a los jugadores que el propósito de la prueba es obtener retroalimentación del comportamiento del juego y enfatizar que se está evaluando el juego y no sus habilidades, no hay respuestas incorrectas y las dificultades que tengan mientras juegan generan información importante para mejorar y evaluar el producto.

  Pedir a los jugadores que piensen en voz alta mientras juegan, para poder escuchar las razones por las que eligen ciertas acciones y tener un panorama más completo de la sesión. En caso de que olvide pensar en voz alta, recordarle de manera amable al preguntar qué es lo que está pensando.

  El tiempo de la sesión de juego debe ser entre 15 y 20 minutos, en caso de que dure más, es posible que se sientan cansados y afecte los resultados de la prueba.
- **Cuestionario de retroalimentación** $(\leq 15\ min)$**:** Al final de la sesión de juego entregar la segunda parte del cuestionario para conocer su opinión del desempeño del juego como la dificultad, los tiempos de carga, la calidad de los niveles jugados, las dificultades que experimentaron para terminar el nivel o si llegó a haber uno imposible de terminar.

El cuestionario se encuentra en el apéndice B, donde las primeras 8 preguntas son de contexto, que deben ser respondidas antes de la sesión de juego, y se busca poder obtener el perfil del jugador, cuáles son sus videojuegos favoritos, cuánto tiempo dedica a la semana a jugar y en qué plataformas prefiere jugar, esto con el fin de conocer la tolerancia que puede llegar a tener ante la frustración o el aburrimiento al jugar videojuegos, como se puede observar en la figura 2.4.

Durante la sesión de juego, existen diferentes causas para darla por terminada, la primera es que se cumpla el tiempo estimado para ésta, sin embargo, hay que tener en cuenta los casos en que algo puede salir mal, como que el jugador pierda todas las vidas y pierda el juego, debido a un nivel muy difícil o imposible de terminar.

Después de la sesión de juego, se da al jugador la segunda parte del cuestionario, las preguntas 9 a 16 del apéndice B, en donde se busca saber el comportamiento del generador de contenido, principalmente en lo referente a las características requeridas en una solución procedural, si le fue posible completar todos los niveles, el diseño, la longitud y el contenido de éstos. El otro punto importante es conocer la dificultad de los niveles jugados con respecto a sus habilidades, por lo que se pregunta si hubo niveles que fueran demasiado fáciles o difíciles, y en caso de que sea positiva la respuesta, se les pide que expliquen la razón.

Teclado	**Control**	**Acción**
→	→	Mover derecha
←	←	Mover izquierda
↓	↓	Agacharse (solo en estado *Super Mario* o *Fire Mario*)
S	3	Brincar
A	4	Correr
A	4	Disparar(solo en estado *Fire Mario*)

FIGURA 5.2: Controles utilizados en *Infinite Mario Bros*.

Es importante que el cuestionario sea breve, concreto y sencillo, ya que uno de gran longitud o con muchas preguntas abiertas puede llevar a la persona a contestar al azar o no terminarlo.

Posteriormente las respuestas son sistematizadas para poder manipularse y observar el comportamiento del generador de contenido con respecto a cada jugador y en general.

5.3 Resultados

En esta sección se analiza el cumplimiento de las características requeridas por una solución procedural, que son velocidad, confiabilidad, expresividad y controlabilidad, en conjunto con la información obtenida del análisis de los cuestionarios dados a los 10 jugadores que participaron en las pruebas de usuario, así como algunas de las opiniones más relevantes respecto al contenido generado, con el fin de observar los cambios de la dificultad respecto a sus habilidades.

Las pruebas fueron realizadas utilizando una computadora portátil con las siguientes especificaciones:

- **Sistema Operativo**: Ubuntu Mate 18.10 x64
- **Procesador**:Intel Atom x7-Z8750, Quad-core a 1.6GHz

- **RAM**: 8 GB

Como dispositivos de entrada se utilizaron teclado y joystick genérico, dado que el *Framework* de *Infinite Mario Bros* no tiene soporte para utilizar un control, se utilizó el programa *rejoystick*[1], el cual permite asignar botones para que se comporten como teclas.

En total se generaron 81 niveles, divididos entre los 10 jugadores que realizaron la prueba.

5.3.1 Propiedades de una solución procedural

Velocidad

De los niveles generados, se puede observar que el tiempo de ejecución para cada nivel tiene un promedio de 0.18 segundos, con un máximo de 0.51 segundos y mínimo de 0.06 segundos, lo cual abarca todo el proceso de ajuste, el algoritmo genético y el uso de la gramática para generar el siguiente nivel.

Estos tiempos de generación son cortos (menor a 1 segundo), por lo que no son perceptibles para el jugador, y no rompen la continuidad del juego.

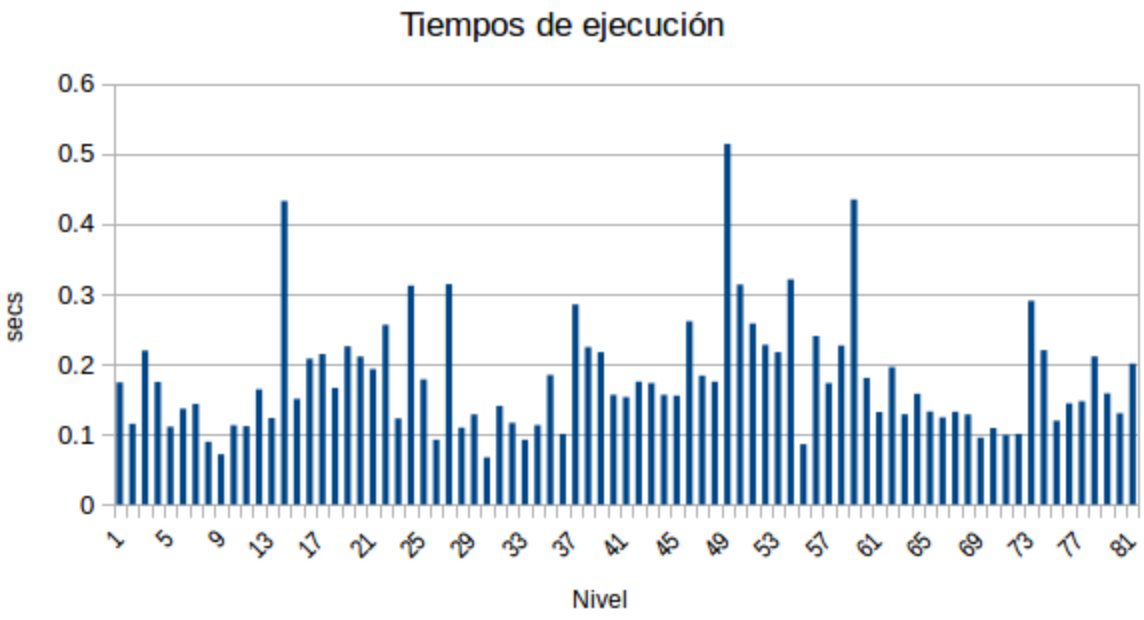


FIGURA 5.3: Tiempos de ejecución del algoritmo generador para todos los niveles de las pruebas.

Es importante notar que los tiempos más altos, son debidos a que el algoritmo genético no logró llegar a un ajuste del 100%, por lo que se realizaron los 100 ciclos del algoritmo genético, en

[1] http://rejoystick.sourceforge.net/

donde se esperaba una linealidad de la estructura con valor de 0.13000001, y el resultado obtenido del mejor individuo de la población fue de 0.13492064 con un valor de ajuste de 0.9999758.

Eliminando los datos atípicos, se puede observar que el tiempo que tarda en obtenerse un resultado que tenga un valor de función de ajuste de 1, con una tolerancia de 0.0001, oscila entre 0.2 y 0.3 segundos.

Confiabilidad

Existen reglas, durante la construcción, que evitan que se puedan generar niveles que sean físicamente imposibles de terminar, sin embargo, dado el comportamiento emergente que muestra el sistema puede pasar, con baja probabilidad, que se generen combinaciones de elementos que impidan completar el nivel o sea muy difícil, como ocurrió durante una de las pruebas, donde al inicio del nivel había una tortuga y un cañón que disparaba una bala en la misma columna, por lo que ambos avanzaban hacia el personaje a la misma velocidad, requiriendo que el jugador realizara un movimiento demasiado complejo.

Además de esto, dos jugadores comentaron que no pudieron terminar el nivel, sin embargo, fue por razones de dificultad, en donde morían fácilmente con precipicios y perdieron todas sus vidas. Estos son los jugadores 6 y 7, donde la dificultad de su último nivel fue de 0.54154575 y 0.540833 respectivamente, y el jugador 7 mencionó que no le gustan los juegos de plataforma y el tiempo que dedica a jugar es cercano cero.

Expresividad

Para medir la expresividad de un nivel, se utiliza la medida de linealidad, que está dada por el resultado del algoritmo genético, en la figura 5.4 se muestran los valores de linealidad de todos los niveles generados en las pruebas, donde se tiene un promedio de 0.17, un máximo de 0.31 y mínimo de 0.03; donde el jugador 5 obtuvo todos los valores menores, ya que evitó realizar saltos lo más que pudo, llevando al sistema a generar niveles con valores cada vez más bajos en la variable de linealidad en cada iteración, llegando a tener niveles totalmente planos.

Sin embargo, descartando a este jugador, se puede observar que la linealidad de los niveles oscila entre 0.10 y 0.25, valores en donde es posible apreciar suficiente variación en las alturas y caminos a elegir para llegar a la meta.

Controlabilidad

Esta función recibe los datos que se generan por el jugador durante un nivel, e impactan directamente en el nuevo nivel, ya que el sistema de ajuste de la dificultad se encarga de analizarlos y modificarlos para que el sistema de juego los muestre al jugador.

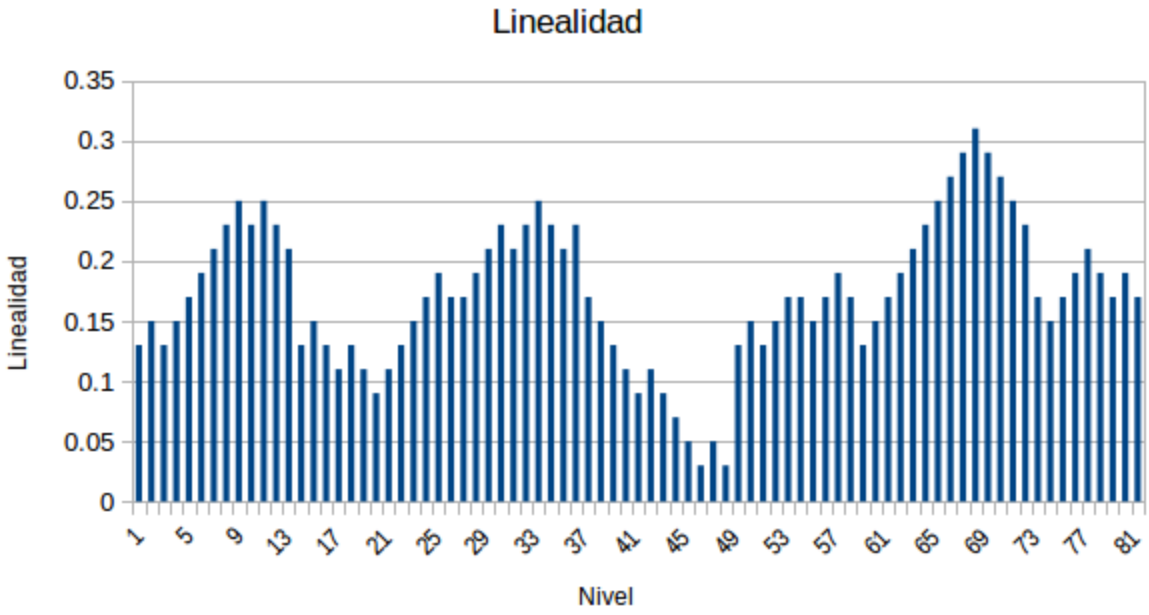


FIGURA 5.4: Valor de linealidad de todos los niveles de las pruebas.

5.3.2 Calificación de los niveles

Una parte importante del diseño centrado en el juego, es realizar pruebas de usuario para determinar si la experiencia de juego es la esperada, ya que los jugadores son quienes van a utilizar el juego.

La calificación que da un jugador a los niveles es subjetiva y depende de variables como qué tipo de jugador es, si le gustan los videojuegos y que géneros prefiere. Esto impacta en la tolerancia que tienen al jugar los niveles que se les presentaron de *Infinite Mario Bros*.

A continuación se describen los resultados obtenidos en los aspectos de diseño, contenido, longitud y dificultad.

Diseño

Diseño se refiere a la forma en la que los elementos que componen el nivel están colocados para generar las experiencias de juego. Al ser generado de manera procedural, se crea sobre estructuras rígidas que tienen algunas modificaciones como efecto del sistema de ajuste de dificultad.

A esto, tres de los jugadores mencionaron que era un buen diseño, seis que era normal y uno que no pudo terminar el primer nivel no pudo contestar esta pregunta.

Contenido

El contenido del nivel, se refiere a los distintos elementos que están dentro de éste, como enemigos, plataformas, bloques, monedas y *powerUp's*.

En la encuesta, después de jugar, tres de los jugadores mencionaron que era un buen diseño, seis que era normal y uno que no pudo terminar el primer nivel no pudo contestar esta pregunta.

Longitud

En este aspecto del juego, se busca conocer si el tiempo definido de 100 segundos para acabar el nivel es adecuado. Durante la sesión del juego, la mayoría de los jugadores mencionó que era largo, con un promedio de 420 unidades, por lo que es necesario reducir esta cantidad de tiempo para tener una longitud menor.

Algunos mencionaron durante la sesión de juego que sería deseable tener puntos de salvado a lo largo del nivel, para en caso de morir, no tener que iniciar desde el principio, mientras que otros mencionaban que debería de darse otro nivel más sencillo en cuanto se perdiera una vida.

Dificultad

Uno de los puntos más importantes de este trabajo es el ajuste de dificultad del contenido generado para cada jugador, por lo que se preguntó en el cuestionario como habían apreciado la dificultad de los niveles jugados, y si le tocó alguno muy difícil o muy fácil.

Los resultados fueron:

Dificultad	**Cantidad**
Muy difícil	0
Difícil	1
Normal	4
Fácil	3
Muy fácil	1

FIGURA 5.5: Dificultad promedio de los niveles jugados.

Sin embargo, la mayoría de los jugadores les tocó un nivel muy fácil, en donde su opinión fue que no había muchos enemigos ni precipicios y había una gran cantidad de monedas.

Por otro lado, en las respuestas a la pregunta sobre un nivel muy difícil, fueron 50% si y 50% no, explicando que la razón por la que si había uno muy difícil era por caídas o saltos difíciles, una gran cantidad de enemigos sobre las plataformas o precipicios repentinos después de una serie larga de plataformas.

Para este caso, se puede observar una correlación entre las personas que mencionaron que les había tocado un nivel muy difícil, son aquellas que dedican muy poco tiempo por semana a jugar videojuegos. Con una excepción, en donde dicha persona dedica 21 horas por semana a jugar, pero menciona que si le tocó un nivel muy difícil y perdió todas sus vidas en un precipicio.

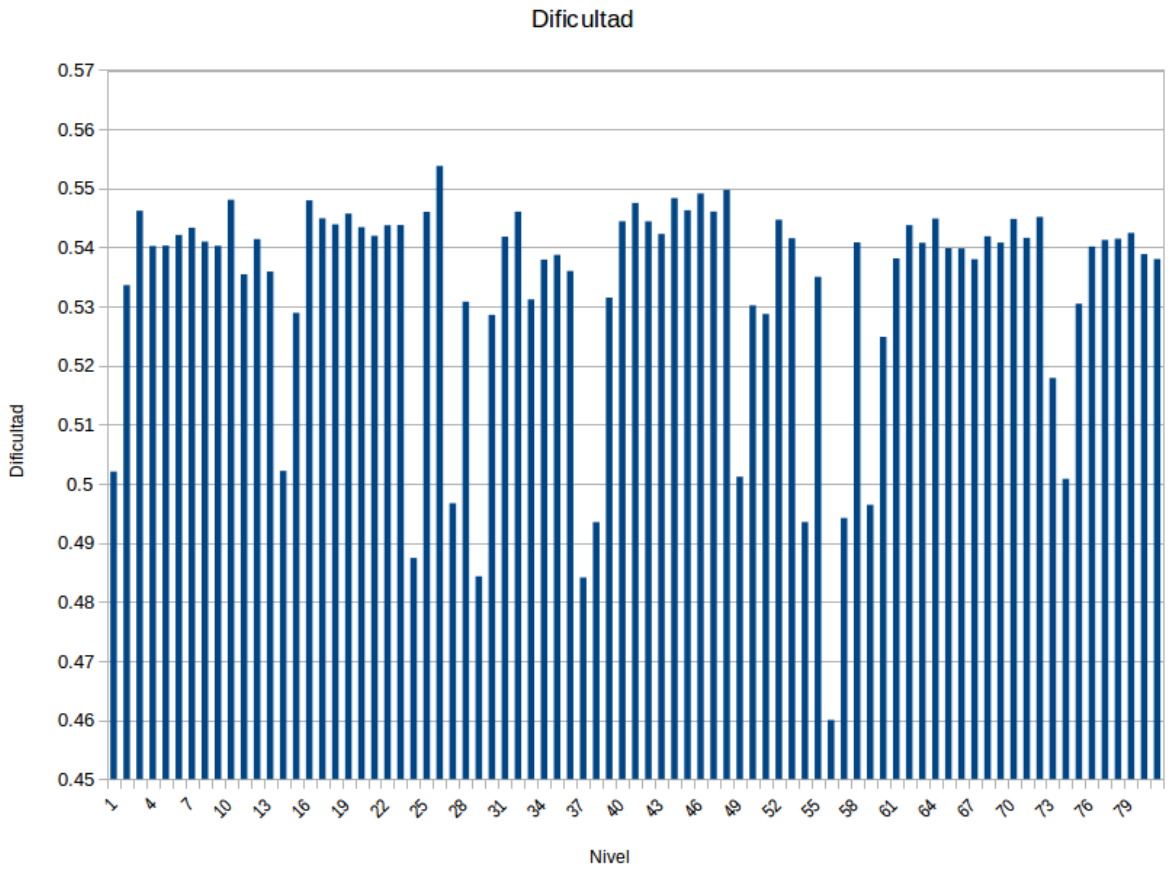


FIGURA 5.6: Valor de dificultad de todos los niveles de las pruebas.

A manera de resumen, se muestra una gráfica con los valores de dificultad, tiempo de ejecución y linealidad de todos los niveles, donde se puede observar que la linealidad y la dificultad son variables no dependientes, ya que puede ser un nivel plano pero de alta dificultad o uno con alto valor de linealidad pero baja dificultad.

Finalmente, en el apéndice C se encuentran las gráficas de los resultados individuales de cada jugador, donde se muestran, de manera similar a la gráfica 5.7, los valores de dificultad, tiempo de ejecución y linealidad.

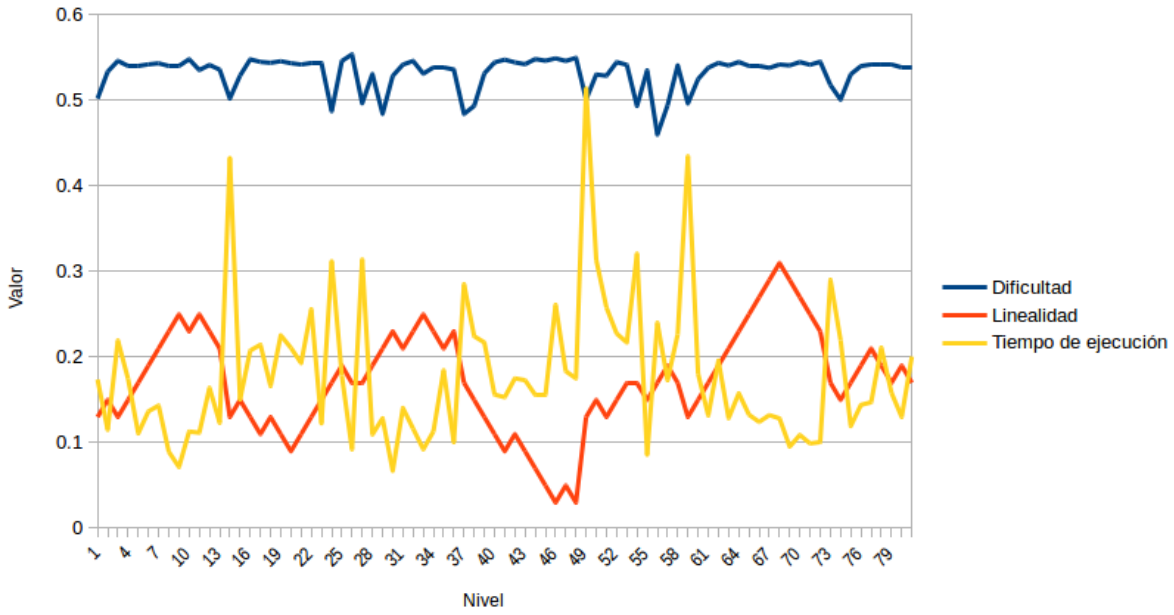


FIGURA 5.7: Valores de dificultad, tiempo de ejecución y linealidad de todos los niveles de las pruebas.

Conclusiones

Este trabajo se centra en la generación procedural de niveles para videojuegos del género de plataformas cuyo contenido ajusta la dificulta a las habilidades del jugador, en donde se utiliza el *Framework* de *Infinite Mario Bros* para mostrar los resultados.

Primero se analiza el diseño y la estructura de un videojuego con el fin de localizar los elementos formales y dramáticos, así como los sistemas de juego que lo componen para poder determinar cuáles son los que tienen impacto en la dificultad del videojuego.

Posteriormente, se analizan diferentes usos de la generación procedural de contenido y, dependiendo de qué tipo, la fase del diseño en que se debe incorporar. También cuáles son las características que una solución procedural debe cumplir para funcionar de manera adecuada.

Adicionalmente, se aborda el tema de *Flujo*, propuesto por *Mihaly Csikszentmihalyi*, quien lo define como la sensación que experimenta una persona al enfocarse completamente en una actividad con un alto grado de satisfacción y los ocho elementos que debe cumplir dicha actividad para lograr esto. Luego, desde el contexto de videojuegos, estos elementos son una premisa, resultados, balance entre reto y habilidades en el contenido y una sensación de control.

Esto nos lleva tener que implementar un sistema de ajustes del videojuego, en donde se propone un ciclo iterativo compuesto de cuatro fases, cada una recibe datos de la anterior y genera un conjunto de datos nuevos para la siguiente, a lo que se define como ajuste pasivo de la dificultad, ya que el jugador debe completar el nivel que esta jugando para poder continuar. Este ajuste inicia con los datos recabados por el sistema cuando se juega un nivel, posteriormente, el sistema de control filtra aquellos que dan información acerca del desempeño del jugador, y el sistema de análisis se encarga de obtener resultados para modificar el contenido de acuerdo a la habilidad del jugador, que posteriormente son enviados al sistema de juego que genera un nivel que cumpla con dichas características.

Se define la jerarquía de un nivel de un juego de plataformas, el cual está compuesto por cápsulas, que dan la estructura, cada una contiene un ritmo, definido por el diseñador, y por último, los elementos más básicos del nivel, como enemigos, bloques, monedas y *powerUp's*.

La implementación consiste de dos fases, la primera, para generar la estructura del nivel se utiliza un algoritmo genético, el cual ofrece soluciones de alta calidad a un problema con la capacidad de adaptarse utilizando una función de ajuste. Primero fue necesario codificar los elementos de la

población a una estructura que permita aplicar los operadores genéticos de cruza y mutación, lo cual se logró con una matriz de valores binarios, donde 1 representa la existencia de una cápsula y, utilizando la vecindad de *Moore*, los 1's adyacentes, representan las cápsulas a las cuales es posible acceder. La función de ajuste mide la linealidad de esta estructura y da mejor calificación a los elementos que se encuentren más cercanos al valor deseado.

En la segunda fase se utiliza una gramática libre de contexto para representar los elementos que componen el nivel, utilizando como estructura el resultado del algoritmo genético, esta gramática coloca cada elemento dentro del nivel, siguiendo los resultados del sistema de análisis. Se debe hacer en énfasis en las reglas para no generar elementos inalcanzables o en el peor de los casos que el nivel no pueda ser completado.

Una vez que el programa estuvo listo, se procedió a realizar pruebas de juego, como del diseño de videojuegos centrado en el juego, para poder obtener datos objetivos del producto, y poder tener información acerca de la dificultad de los niveles generados.

Durante las pruebas se observó que los tiempos de ejecución eran rápidos, tardando menos de un segundo por cada nivel generado, lo que permite jugar de manera continua con tiempos de carga imperceptibles para el jugador.

En cuanto a la dificultad, se tuvieron jugadores con diferentes perfiles, con respuestas variadas, desde normal a muy fácil y solo uno en difícil, y la mayoría mencionó que el promedio era normal. Sin embargo sí hubo diferentes respuestas en cuanto a la existencia de un nivel con una dificultad muy baja o muy alta.

En las gráficas de cada persona, de dificultad, linealidad y tiempo de cada nivel (apéndice C), se puede observar como el valor de dificultad se ajusta gradualmente, pero cuando el personaje pierde una vida cae de manera brusca y comienza a subir si supera los siguientes niveles. Es en dichas caídas, es cuando las personas mencionaron que les tocó un nivel muy fácil.

Debido a esto es necesario tener en cuenta un mayor número de variables para tener un mejor ajuste, ya que surge la problemática de qué tan grandes deben ser las variaciones. Si son grandes, es posible que se presenten niveles muy difíciles en una etapa temprana de la sesión de juego, pero si son muy pequeñas, después de perder y que el sistema baje el nivel de forma drástica, tardará mucho en llegar de nuevo a una dificultad adecuada.

Otro aspecto importante observado durante las pruebas fue que debido tenían otras actividades que realizar, después de terminar los 15 minutos requeridos, no les era posible continuar jugando, por lo que, para trabajos futuros de este tema, será necesario realizar pruebas con un ambiente más controlado y buscar disponer de más tiempo por cada evaluado.

En cuanto a la confiabilidad del contenido generado, todos los niveles estaban compuestos por plataformas que no tenían precipicios físicamente imposibles de superar, y siempre existía un camino del inicio a la meta, sin embargo, debido al comportamiento del generador, hubo un nivel

en donde dos enemigos se sincronizaban y hacía que fuera demasiado difícil evitarlos, por lo que es necesario realizar una validación de este tipo de casos para que no suceda.

Es importante mencionar que deben ser agregados más elementos dramáticos al juego y la generación procedural no debe ser el único atractivo del juego, ya que es importante generar una conexión con el jugador para que éste desee jugar una cantidad de niveles virtualmente infinita o buscando tener alguna recompensa adicional en la historia del juego. Esta conexión es importante, ya que si no tiene una razón para jugar, no habrá concentración en la tarea, lo que es un elemento requerido para que una persona llegue a un estado de *Flujo*.

Finalmente, aunque el videojuego de *Infinite Mario Bros* fue utilizado para mostrar este trabajo, es posible implementar esta solución procedural a otros del mismo género, ya que con lo que se trabajó fueron los elementos formales y sistemas de juego, los cuales son compartidos por diferentes juegos del mismo género.

Apéndice A

Una síntesis del diseño y estructura de un videojuego

El diseño de un juego consiste en llevar un proceso iterativo en el cual se colocan las metas de experiencia de juego, generan ideas o el sistema, formalizan de manera escrita o con un prototipo, realizan pruebas de usuario para comprobar la funcionalidad y evaluar los resultados para posteriormente llevar a cabo las modificaciones pertinentes y de ser necesario repetir el ciclo en cada fase hasta lograr resultados positivos.

Un juego es un sistema que se compone de elementos formales, que definen la esencia de éste y dramáticos, que dan contexto al juego integrando los elementos formales y dan significado a la experiencia de juego[4].

A.1 Proceso de diseño centrado en el juego

Un concepto clave para pensar como diseñador de videojuegos es tener un proceso sólido para desarrollar una idea, desde el concepto inicial hasta el producto terminado, es decir, una versión jugable para el público objetivo. Esto significa mantener como prioridad que la experiencia de juego sea la esperada en cada fase de desarrollo del videojuego, por lo que es de gran importancia incluir al jugador dentro del proceso lo más pronto posible[4].

A.1.1 Metas de experiencia de juego

Una meta de experiencia de juego es una configuración de los elementos que generan una situación única dentro del juego, la cual se espera que el jugador encuentre por si mismo. Cabe mencionar que la experiencia de juego no es una característica del mismo, es lo que se espera lograr utilizando los elementos desarrollados para éste.

Colocar la experiencia de juego como parte del proceso creativo permitirá enfocar los esfuerzos al desarrollo de características que ayuden a lograr las metas de experiencia de juego, por lo que una pieza clave de esta fase es la retroalimentación generada por el *playtest*[1], lo que permitirá observar de manera inmediata y objetiva si las metas de experiencia de juego son alcanzadas. Se deberá poner mayor atención a lo que experimenta el jugador, la razón por la que tomó dichas decisiones durante el juego, qué lo llevó a tomarlas y si logró completar el objetivo de la manera esperada[4].

A.1.2 Prototipo y pruebas

Otro componente clave en el diseño centrado en el juego es la necesidad de crear un prototipo de las ideas a desarrollar y ser probado lo más pronto posible. Inmediatamente después de proponer ideas para el desarrollo se debe hacer una versión jugable de las mismas, la forma más utilizada es crear un prototipo físico que permita mostrar las mecánicas del núcleo del videojuego.

La meta de esta actividad es poder jugar y perfeccionar el modelo antes de incluirlo en el proyecto. De esta manera el diseñador recibe retroalimentación inmediata de lo que piensan los jugadores y observar si dicha idea ayuda a lograr las metas de experiencia de juego, así como a encontrar fallas en la mecánica o núcleo del videojuego.

Comenzar a desarrollar el videojuego sin haber perfeccionado tanto el núcleo como las mecánicas del videojuego para lograr las experiencias de juego deseadas, incrementará el esfuerzo requerido para modificar partes del código conforme se avance en el desarrollo, lo cual puede derivar en gastos de tiempo y esfuerzo adicionales y en el peor de los casos que el proyecto no pueda ser concluido en tiempo y forma[4].

A.1.3 Proceso iterativo

El proceso iterativo consiste en las fases de diseño, prueba y evaluación de los resultados continuamente durante el desarrollo del videojuego, y en cada ciclo mejorar las mecánicas del juego o sus características hasta que las experiencias de juego cumplan los objetivos deseados. Llevar a cabo este proceso es una pieza fundamental en el desarrollo de un videojuego con diseño centrado en el mismo[4].

A continuación se describe el flujo de las fases que debe seguir el proceso iterativo (figura A.1):

- Colocar metas de experiencia de juego.
- Generar una idea o sistema.

[1] Proceso en el que se presenta un juego en fase de desarrollo a una audiencia específica para identificar posibles fallas y obtener retroalimentación.

- Formalizar la idea o sistema, ya sea de manera escrita o creando un prototipo.
- Probar la idea o sistema, utilizando jugadores o presentándola.
- Evaluar y dar prioridad a los resultados.
- Si los resultados no son convincentes, regresar al primer paso.
- Si los resultados indican que hay elementos que mejorar, modificar y regresar al paso uno.
- Si los resultados son positivos o convincentes, el proceso de iteración ha sido concluido.

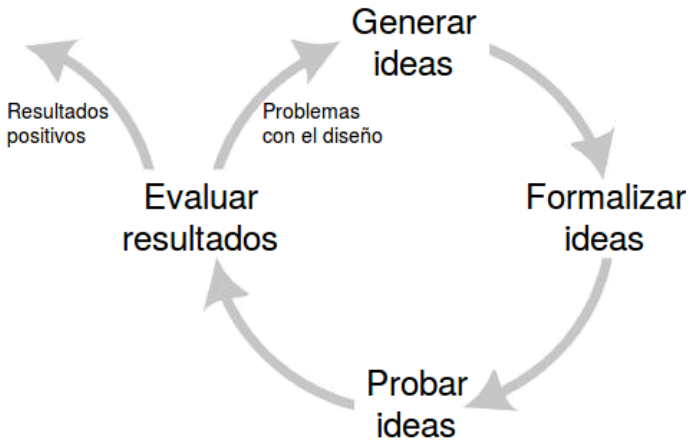


FIGURA A.1: Diagrama de las fases que componen el proceso iterativo (Figura adaptada de [4]).

Como se puede observar, el proceso iterativo puede ser aplicado en cada fase del desarrollo del videojuego, desde el concepto inicial hasta las pruebas de calidad del sistema.

Fase 1: Lluvia de ideas

- Colocar metas de experiencia de juego.
- Presentar ideas o mecánicas de juego que puedan ayudar a lograr las metas de experiencia.
- Escribir un breve documento que incluya la descripción de de las ideas.
- Realizar una prueba de las ideas utilizando algún apoyo visual para facilitar la comunicación de las ideas hacia algunos candidatos que pertenezcan a la población objetivo.

Fase 2: Prototipo físico

- Crear un prototipo jugable sencillo, utilizando papel y lápiz u otros materiales sencillos y de bajo costo.
- Probar el prototipo con algunos jugadores potenciales.

- Cuando el prototipo físico muestre que con los elementos incluidos es posible lograr los objetivos de la experiencia de juego, escribir un documento detallado de las mecánicas del juego.

Fase 3: Presentación

- Una de las razones principales para crear una presentación es conseguir fondos para desarrollar el videojuego, sin embargo, aunque ésto no sea necesario, crearla permitirá facilitar compartir ideas a todo el equipo de trabajo y así asegurar que todos tendrán el mismo enfoque.
- La presentación debe tener una demostración conceptual del arte y una descripción detallada de las mecánicas del juego.
- La presentación debe ser hecha gastando la menor cantidad de recursos posible, así, en caso de que no cumpla con las metas esperadas, se puede regresar a la fase 1 y comenzar con un nuevo proyecto o realizar los cambios pertinentes.

Fase 4: Prototipo de software

- Hacer varios prototipos para exhibir diferentes aspectos del núcleo del videojuego.
- Realizar el prototipo con gráficas temporales, para reducir al máximo los costos de desarrollo y lograr avances mas rápidos.
- Realizar pruebas sobre el prototipo desarrollado con los *playtesters*.
- Cuando el prototipo muestre que se logran cumplir las metas de experiencia de juego se puede proceder a la siguiente fase.

Fase 5: Documentación de diseño

- Utilizar las notas creadas durante la etapa de desarrollo del prototipo permitirán facilitar escribir el primer borrador que describa cada aspecto del videojuego y la manera en como funciona.
- Otra forma de crear este documento es la generación de una *wiki*[2], que es una herramienta que facilita la colaboración entre todo el equipo y permite que el documento se vaya complementando conforme el desarrollo del videojuego avanza.

Fase 6: Producción

- Trabajar con todos los miembros del equipo para asegurar que las metas plasmadas en el documento de diseño son alcanzables y que están descritas de manera correcta.
- Esta fase puede comenzar después de concluir el primer borrador del documento de diseño.

[2]Comunidad virtual, cuyas páginas son editadas directamente desde el navegador, donde los mismos usuarios crean, modifican, corrigen o eliminan contenidos que, normalmente, comparten.

- El propósito de esta fase es comenzar a crear el arte del videojuego y la programación del mismo.
- Como parte del diseño centrado en el videojuego es muy importante no dejar de lado el proceso de probar y evaluar cada aspecto desarrollado del videojuego, arte, mecánicas del juego, personajes, etc. Siguiendo este proceso los problemas que se vayan encontrando serán cada vez menores, ésto es debido a que lo más complicado ya fue resuelto durante las fases anteriores.

Paso 7: Calidad

- Para iniciar este paso se debe tener una sólida mecánica de juego. Aún es posible que existan algunos problemas, así que el proceso iterativo debe seguir aplicándose, enfocado en mejorar la usabilidad.
- Se debe asegurar que el videojuego es accesible a la población objetivo.

Debido a los grandes avances y competencias en la industria de los videojuegos, los diseñadores de videojuegos tendrán que crear nuevas y diferentes experiencias de juego, por lo que deberán tomar mayores riesgos para innovar[4]:

- Diseñar videojuegos con mecánicas de juego únicas, más allá de los géneros actualmente existentes.
- Incluir a poblaciones diferentes, personas que tengan diferentes gustos o habilidades.
- Abordar desde diferentes ángulos los problemas en el diseño de videojuego, como:
  - Integración de juego e historia.
  - Mayor empatía entre los personajes dentro del videojuego.
  - Crear videojuegos con más profundidad en las características de los elementos.
  - Descubrir relaciones entre videojuegos y aprendizaje.
- Observar el impacto que puede generar personal y culturalmente.

El diseño centrado en el videojuego ofrece un proceso de desarrollo sólido durante todas sus fases, el cual permite explorar diferentes posibilidades para las mecánicas del juego, tener retroalimentación por parte de los *playtesters* durante todo el proceso de cada una, permitirá desarrollar ideas de manera robusta manteniendo siempre la experiencia de juego como parte de los objetivos a alcanzar[4].

A.2 Estructura de un juego

En la actualidad existen diversos géneros de videojuegos, cada uno con características y requerimientos particulares, sin embargo todos son considerados juegos, ésto es debido a que comparten algunos elementos que muestran comportamientos diferentes de acuerdo a su propio diseño y mecánicas de juego[4].

Estos elementos se clasifican en formales y dramáticos. Los elementos formales son aquellos que son compartidos por todos los juegos, estos son la esencia y dan forma a un juego. Mientras que los elementos dramáticos permiten al jugador involucrarse emocionalmente con la experiencia de juego, esta sensación varía de acuerdo a cada jugador.

A continuación se describen los elementos formales, aquellos que son compartidos entre diferentes juegos. La ausencia de jugadores, objetivos, reglas, recursos, conflicto, retro o resultados, evitaría que se tuviera la estructura mínima para que el sistema pueda ser definido como un juego[4].

- **Jugadores**: La principal similitud es que ofrecen experiencias diseñadas para jugadores, la cual es diferente a otras formas de entretenimiento, ya que demandan la participación activa de los consumidores. A diferencia de la música, en donde los músicos crean la experiencia musical y los consumidores son las personas de la audiencia.

  La forma en la que los jugadores son requeridos en el juego varía de acuerdo al género del mismo, en el videojuego *busca-minas*[3] el diseño requiere de solo un jugador para comenzar el juego, mientras que en el videojuego *Mortal Kombat*[4] se requieren de uno a 4 jugadores. Aunque los escenarios son totalmente diferentes, el término **jugador** implica voluntariamente participar y disfrutar del entretenimiento.

  Para ser un jugador, se deben aceptar las reglas y restricciones del juego. Esta aceptación es parte de los estados emocionales y psicológicos que se deben considerar como parte del proceso de diseño centrado en el juego.

- **Objetivos**: Otro elemento que comparten es el tener objetivos bien definidos. En el *busca-minas* el objetivo es descubrir todas las casillas que no contienen minas, mientras que en *Mortal Kombat* la meta consiste en terminar con la barra de puntos de golpe y derrotar al contrincante.

  En los juegos el objetivo es un elemento clave, sin éste se perdería la estructura de la experiencia de juego, lo cual indicaría una error durante el proceso de diseño centrado en el juego.

[3] 1989, Robert Donner. El objetivo del juego es despejar un campo de minas sin detonar ninguna.
[4] 2013, Warner Bros. Interactive Entertainment. Un videojuego de peleas en 2D.

- **Procedimientos**: Son descripciones detalladas de cómo los jugadores pueden lograr cumplir los objetivos del juego dentro del conjunto de reglas definidos por el mismo. Esto incluye los controles, que son métodos a través de los cuales es posible interactuar con el videojuego. Por ejemplo, en el videojuego *busca-minas* se juega una casilla a la vez, se elige entre descubrir para ver el contenido o marcar como mina, mientras que en *Mortal Kombat* el personaje puede avanzar, retroceder, defender, atacar (golpes, patadas o movimientos especiales) y brincar.

- **Reglas**: Sirven para describir exactamente en qué consisten los objetos dentro del juego y qué es lo que los jugadores pueden y no hacer. También muestran las acciones que se pueden realizar en situaciones particulares.

  Algunas de las reglas sirven para definir objetos y conceptos mientras que otras sirven para limitar las acciones del jugador.

  La reglas y los procedimientos son formas de autoridad, que implícitamente los jugadores aceptan para ser parte del juego. Si no se siguen las reglas, no se estaría jugando realmente ese juego, esto significa que los elementos estructurales estarían siendo ignorados y sería difícil lograr la meta de experiencia deseada.

  Por ejemplo, en el videojuego *busca-minas* solo puedes descubrir una casilla o marcarla como mina y una vez que se descubre una mina el juego termina. Mientras que en *Mortal Kombat* en peleas a tres rondas, gana el primero en vencer en dos rondas.

- **Recursos**: Estos pueden tener diferentes valores para los jugadores en la búsqueda de cumplir el objetivo. Tienen valor porque ayudan al jugador a alcanzar su objetivo, pero son limitados en el sistema por el diseñador.

  Encontrar y administrar de manera adecuada los recursos es una parte fundamental de algunos juegos, por ejemplo cartas, armas, tiempo, terreno, etc..

  Los recursos, por definición, tienen valor debido a su utilidad y cantidad en el juego. Algunos se pueden combinar con otros para generar nuevos objetos o intercambiar por otro tipo de objeto con diferente utilidad.

  Por ejemplo, en el videojuego *busca-minas* los recursos son las banderas para marcar las casillas que podrían ser minas, pero si se marca alguna que no lo es, las banderas no alcanzarán. Mientras que en *Mortal Kombat* un recurso es el tiempo del combate, que si se agota, el ganador será decidido por quién tenga más puntos de combate.

- **Conflicto**: Surge a partir de los procedimientos y reglas determinadas dentro del juego limitando las acciones del jugador, en la búsqueda de resolver un objetivo utilizando los recursos existentes.

Por ejemplo, en el videojuego *busca-minas* se busca descubrir todas las casillas evitando las minas en el menor tiempo posible. Mientras que en *Mortal Kombat* se busca acabar con la barra de puntos de combate del oponente antes de perder todos los puntos de combate de el personaje utilizado.

- **Resultados**: Debido a las reglas y restricciones determinadas en cada juego, los resultados son inciertos, pero es posible medirlos, ganar, perder, empate, clasificación, etc.

  El resultado del juego difiere del objetivo, pero existen otros factores que pueden afectar la puntuación y determinar quién es el ganador del juego, o la posición que tendrán en la clasificación.

  Es importante el factor de incertidumbre en el resultado, ya que es un motivador para los jugadores. Si el jugador puede anticipar el resultado del juego, dejará de jugarlo.

  Por ejemplo, en el videojuego *busca-minas* si se descubre una casilla que contiene una mina, el juego termina y el jugador pierde, pero si se descubren todas las casillas que no son minas el jugador gana. Mientras que en *Mortal Kombat* cuando el personaje contrario queda sin puntos de golpe, el jugador gana el combate.

A continuación se listan algunos de los elementos dramáticos que permiten generar una conexión emocional entre el jugador y el juego[4]:

- **Reto**: Los juegos crean un conflicto en el que los jugadores deben realizar ciertas acciones para superarlo. Esto genera tensión durante la experiencia de juego, la cual varía entre diferentes niveles de logro y frustración. Incrementar el reto mientras se avanza en el juego puede causar un aumento en la sensación de tensión, si el reto es muy difícil puede causar frustración, pero si se mantiene igual o disminuye de manera considerable, los jugadores pueden tener la sensación de haber dominado el juego y dejarlo.

- **Juego**: La relación entre juegos y juego tiene un papel importante, ya que jugar en si no define al juego. Jugar se define como *movimiento libre dentro de una estructura rígida*[5]. El juego existe gracias a los elementos formales que proveen las estructuras necesarias para que el jugador tenga la oportunidad de usar su imaginación, inspiración y la libertad para completar los objetivos dentro del juego.

- **Premisa**: Una forma básica de involucrar al jugador es una premisa general, que da contexto a los elementos formales. Adicionalmente es una herramienta que facilita involucrar a los jugadores emocionalmente en su interacción con los elementos formales.

- **Personaje**: Con la llegada de los juegos digitales, surgió otra herramienta para involucrar al jugador, que es la noción de personaje. Estos permiten ser un *avatar*[6] para que el jugador

[5] Salen and Zimmerman, [4]

[6] Es la representación gráfica de un usuario dentro de un videojuego que permite interactuar con lo elementos de éste.

participe dentro del juego, experimente diferentes situaciones y conflictos mientras avanza a través de la historia.

- **Historia**: Algunos juegos crean una historia utilizando los elementos formales para involucrar emocionalmente aún más al jugador. A diferencia de la premisa, la historia se desarrolla conforme se avanza en el juego. La forma en la que se integra una historia al juego puede variar ampliamente, pero claramente, tanto para los jugadores y diseñadores, puede generar resultados emocionales profundos.

Los juegos son sistemas, debido a las relaciones formadas entre todos los elementos, generando algo más complejo cuando todo es unido. Es posible entender y predecir como se comportarán las funciones en la interacción con otros elementos, pero cuando se unen todas las partes y funciona como un todo, se pueden apreciar importantes cualidades emergiendo como consecuencia de la interacción del jugador con los elementos del juego[4].

Por lo que un juego se define como un sistema dinámico, donde los elementos formales dan estructura al juego, mientras que los elementos dramáticos aportan lo necesario para generar experiencias emocionales, así como generar un resultado concreto y posible de medir para mostrar a los ganadores o una clasificación de éstos.

A.3 Elementos formales

Los elementos formales son aquellos que definen la estructura de un juego, sin los cuales dejaría de ser considerado un juego. Para diseñar un juego es necesario comprender cuáles son los elementos esenciales que lo componen así como las relaciones que existen entre estos. La esencia de un juego son los jugadores, objetivos, reglas, recursos, conflictos y resultados[4].

A.3.1 Jugadores

Los juegos son experiencias diseñadas para los jugadores, los cuales deben aceptar voluntariamente las reglas y restricciones definidas por éste para poder jugar. Una vez aceptado los términos, podrán realizar acciones que nunca se harían en otras situaciones, como disparar, matar o traicionar a otros personajes. Por otra parte también es posible realizar ciertas acciones bajo ciertas situaciones ficticias en las que el jugador tendrá la oportunidad de realizar o enfrentar, como sacrificarse por algo o alguien, tomar una decisión difícil o manejar un vehículo aéreo. Estas acciones, dentro de los límites y reglas del videojuego, permiten al jugador la oportunidad de llevarlas a cabo sin repercusiones en el mundo real[4].

Invitación a jugar

Uno de los momentos más importantes es la invitación a jugar. En un tablero o juego de cartas, la invitación es por parte de la sociedad, los jugadores invitan a otros, mientras que en un juego digital el proceso es más técnico, generalmente por una pantalla de inicio o un botón de *iniciar*. Algunos juegos realizan un esfuerzo adicional para ofrecer una invitación más intuitiva, por ejemplo, *Guitar Hero*[7], cuyo control es una guitarra, y cuando el jugador comienza a utilizarla le permite actuar como un guitarrista, donde más que jugar, juega la fantasía del juego. Crear una invitación a jugar, hacerla más intuitiva e interesante para la población objetivo es una parte importante en el diseño centrado en el juego.

Numero de jugadores

Es importante tener en cuenta desde el inicio del proceso de diseño el número de jugadores que requiere el juego para funcionar de manera correcta. Algunos juegos están diseñados para que el jugador compita contra el sistema, mientras que otros requieren un número exacto de jugadores para funcionar. Por otra parte, existen videojuegos diseñados para ser jugados por miles de jugadores simultáneamente, pero al mismo tiempo permiten jugar de manera individual y aún así algunos elementos formales seguirán funcionando de manera adecuada.

Roles de jugadores

Los roles están definidos dentro de las reglas del juego, por lo que deben ser considerados desde el proceso de diseño del videojuego. Los roles definen algunas de las habilidades básicas del jugador. Algunos juegos limitan los roles a los jugadores, como el ajedrez, en el cual solo se puede elegir utilizar uno de los dos colores disponibles para jugar. Mientras que otros juegos ofrecen al jugador la posibilidad de crear personajes con características personalizadas, desde su función en los combates hasta la raza, género y vestimenta que tendrá, lo cual da la sensación de tener una segunda vida en un mundo virtual en donde puede interactuar con otras personas, sin embargo es importante balancear los roles existentes para no dar ventaja a alguna cualidad en particular.

FIGURA A.2: Creación de un personaje en *Skyrim: The Elder Scrolls Online* (Figura tomada de https://cellcode.us/quotes/skyrim-character-sliders-nord.html).

[7] https://www.guitarhero.com/mx/es/game

Patrones de interacción entre jugadores

Otra elección durante el diseño del juego es la estructura que seguirá la interacción entre el jugador, el sistema y otros jugadores. Las principales estructuras definidas son:

1. **Jugador contra el juego** - Debido a que no hay otros jugadores humanos en esta estructura, para generar el conflicto se utilizan *puzzles* o estructuras similares (figura A.3).

FIGURA A.3: Jugador contra el juego (Figura tomada de [4]).

2. **Varios jugadores contra el juego** - Este patrón esta orientado a fomentar la cooperación, en el que un jugador, junto con otros jugadores trabajan de manera conjunta para derrotar al sistema. También se puede observar el caso particular que sería como la estructura número uno, en el que solo un jugador compite contra el sistema (figura A.4).

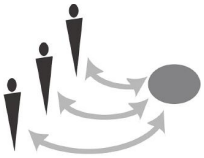


FIGURA A.4: Varios jugadores contra el juego (Figura tomada de [4]).

3. **Jugador contra jugador** - En esta estructura, un jugador compite directamente contra otro (figura A.5).

FIGURA A.5: Jugador contra jugador (Figura tomada de [4]).

4. **Un jugador contra varios jugadores** - Uno o más jugadores trabajan juntos para derrotar a un jugador. En estos casos es necesario tener en cuenta el mantener un balance entre ambas partes de la competencia (figura A.6).

5. **Todos contra todos** - Se requiere de al menos tres jugadores, los cuales compiten directamente. En un caso particular, puede caer en similitud con la estructura cuatro por un tiempo, cuando algunos los jugadores crean alianzas para eliminar a un jugador en particular (figura A.7).

FIGURA A.6: Un jugador contra varios jugadores (Figura tomada de [4]).

FIGURA A.7: Todos contra todos (Figura tomada de [4]).

6. **Cooperativo** - Requiere de más de un jugador, en el cual una pieza clave es la interacción de los jugadores para resolver el conflicto. Algunos diseños permiten que solo un jugador pueda jugar, sin embargo, para lograr la experiencia de juego deseada son requeridos más jugadores (figura A.8).

FIGURA A.8: Cooperativo (Figura tomada de [4]).

7. **Competencia por equipos** - Dos o más grupos se enfrentan. Esta estructura es utilizada en juegos en línea de grandes cantidades de jugadores, en donde se forman clanes para enfrentarse contra otros (figura A.9).

A.3.2 Objetivos

Define qué es lo que los jugadores deben completar para lograr avanzar en el juego dentro de las reglas. Cuando el objetivo es diseñado correctamente, permite generar un reto dentro del juego así como el género del mismo, por ejemplo, uno cuyo objetivo sea eliminar a otro jugador será diferente de uno donde el objetivo sea completar el nivel en el menor tiempo posible[4].

Algunos juegos permiten a los jugadores tener diferentes objetivos o seleccionarlos. Adicionalmente existen objetivos secundarios u objetivos que permitirán eventualmente completar uno más complejo o ayudarán a completar el objetivo principal.

Además de ser un elemento formal, tiene la capacidad de afectar aspectos de los elementos dramáticos, por lo que es necesario poner atención especial para integrar de maneara adecuada el objetivo con la premisa del juego.

FIGURA A.9: Competencia por equipos (Figura tomada de [4]).

A continuación se definen algunas de las categorías en las que se clasifican los tipos objetivos:

1. **Captura:** Consiste en conquistar o destruir algo que el oponente posee mientras se evita perder los objetos propios.
2. **Persecución:** Consiste en perseguir a un oponente o esquivarlo, dependiendo del rol que se juega.
3. **Carrera:** Consiste en llegar a una meta antes que todos los oponentes o en el menor tiempo, dependiendo de la estructura de interacción entre los jugadores.
4. **Alineación:** Consiste en alinear los objetos del juego en una configuración particular definida por las reglas del juego.
5. **Rescate o escape:** Consiste en conseguir que un personaje u objeto del juego logre estar a salvo, a través de retos o niveles que deben ser completados.
6. **Construcción:** Consiste en construir, mantener y administrar recursos dentro de un ambiente directa o indirectamente competitivo. Considerado en algunos casos similar a la categoría de alineación pero a un grado más sofisticado y complejo.
7. **Exploración:** En esta categoría se deja al jugador libre para explorar un área del juego. Usualmente se complementa con otro objetivo como encontrar algún objeto.
8. **Solución:** El objetivo en un juego de solución es resolver un problema o un *puzzle* antes que el contrincante o en un tiempo dado.

Como se mencionó anteriormente, esta lista solo menciona algunas de las categorías en las que se clasifican los objetivos de los juegos. Algunos mezclan varios tipos de objetivos dentro de su mecánica de juego.

A.3.3 Procedimientos

Los procedimientos son los métodos y acciones que tienen los jugadores para completar los objetivos del juego[4].

En los juegos de mesa, los procedimientos están descritos en las reglas del juego que generalmente se encuentran escritos en un documento o se transfieren verbalmente por los

jugadores. Por otra parte, en los juegos digitales también es necesario describir la función de los controles para interactuar con el juego, pero se tienen más medios para realizar esta actividad, como tutoriales o de manera discreta conforme se avanza.

Los juegos digitales pueden llegar a tener un gran número de estados y con mayor complejidad que los demás tipos de juegos. También poseen estados que dependen de procedimientos que se ejecutan en el *background* del sistema, llamados procedimientos del sistema, respondiendo a las acciones del jugador de1 manera automática e inmediata, ofreciendo una experiencia de juego más fluida. Cabe mencionar que esto no significa que los juegos digitales necesariamente son más complejos.

Durante la fase de definición de los procedimientos para el juego, es necesario tener en mente las limitaciones del contexto en el que se utilizará. Los procedimientos están directamente afectados por las limitaciones físicas, por lo que, al diseñar este tipo de elemento, es necesario poner especial atención en que las personas de la población a la que va dirigido el juego puedan entender y recordar dichos procedimientos.

A.3.4 Reglas

Como se ha mencionado, las reglas definen objetos del juego así como las acciones permitidas para los jugadores. Así como los procedimientos, las reglas están descritas en un documento, si embargo, en los juegos digitales también pueden estar implícitas en las mecánicas del juego, ya que el sistema será el encargado de validar las mismas de manera automática y mostrando solo los comandos posibles en un estado[4].

Se debe tener en cuenta la población a la que va dirigido el juego al asignar las reglas, ya que una gran cantidad podría hacer que el juego sea difícil de llevar a cabo o entender, lo que rompería con la fluidez del juego.

Objetos

Los objetos en los juegos tienen un estado único y un significado, definidos como parte de las reglas, los cuales pueden tener algunas similitudes con objetos reales, sin embargo solo son abstracciones, por lo que deben ser definidos por las reglas del juego para dar sentido a éstos.

Los juegos de mesa, así como los juegos que no son digitales, deben definir los objetos explícitamente dentro de las reglas. Estos objetos, dadas las limitaciones físicas, solo poseen una pequeña cantidad de estados, usualmente el aspecto es una herramienta para facilitar su uso.

Por otra parte, los juegos digitales pueden tener objetos como personajes que poseen una gran cantidad de estados y variables que definen un estado general del objeto. Los jugadores no

necesariamente ven este conjunto de variables, dado que el sistema se encarga de manejar los estados automáticamente.

Limitar acciones

Las acciones que pueden llevar a cabo los objetos dentro del juego deben estar definidas dentro de las reglas. Estas pueden ser dependientes del estado en el que se encuentre el objeto, cumplir requisitos previamente definidos como objetivos, tener los recursos requeridos o mantenerse dentro de los límites del juego.

Efectos

Algunas reglas pueden desencadenar una lista de acciones al cumplirse ciertas condiciones particulares. Esto permite crear variaciones del juego, ya sea para penalizar o compensar al jugador por realizar ciertas acciones. Es importante tener en cuenta el balance de los efectos, ya que en caso de crear un impacto muy grande en el juego, podría hacer que el jugador pierda el interés, ya sea debido a que dificulte o facilite de manera drástica el nivel del juego.

Sin embargo también permite llevar al jugador a prestar atención especial en lo requerido para lograr la experiencia de juego definida en el diseño del mismo.

Definir reglas

Al igual que los procedimientos, las reglas se verán afectadas por el entorno donde se juega. Necesitan ser claras e intuitivas, pensando en la población a la cual va dirigido el juego. Entre más difíciles de entender por los jugadores, será menos probable que se logre tener la experiencia de juego deseada y se dará la sensación de tener poco control del juego así como realizar elecciones sin sentido.

A.3.5 Recursos

Al igual que en el mundo real, los recursos en un juego pueden ser utilizados para completar algunas metas. Durante el diseño es importante tener en cuenta la forma en que el jugador podrá obtener y administrar los recursos.

Deben tener principalmente dos características, usabilidad y cantidad dentro del juego. Si los recursos fueran abundantes tendrían poco valor, sin embargo, algunos recursos pueden carecer de usabilidad, permitiendo ofrecer experiencias adicionales con valor dado por el jugador[4].

Otra parte fundamental de los recursos es determinar la forma en la que el jugador los obtiene y administra, así como las limitantes de la cantidad de éstos.

A continuación se listan algunos ejemplos de recursos:

Vidas

Las vidas son un recurso de gran valor en juegos de acción[8]. En juegos como *Super Mario Bros* (figura A.10) y *Spaceinvaders*, llegan a ser el principal recurso, las cuales son necesarias para completar las metas y en caso de perder todas se tiene que comenzar desde el inicio.

También es importante definir en las reglas la forma en que se puede ganar una vida extra, el límite máximo que un jugador puede tener y en qué casos se pierde una.

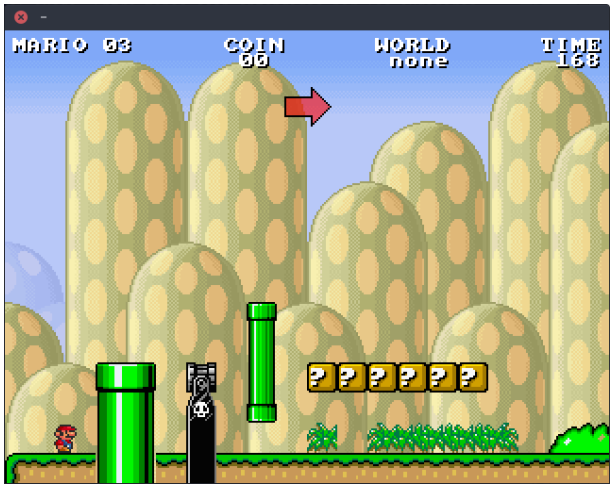


FIGURA A.10: *Super Mario Bros*: tres vidas restantes (se muestra en la parte superior izquierda).

Unidades

En juegos donde el jugador representa más de un objeto al mismo tiempo, generalmente se tienen que administrar las unidades disponibles. Pueden ser de diferentes tipos, en algunos juegos como las *Damas Inglesas* donde solo hay un tipo de unidades o en el *Ajedrez* donde hay mayor variedad de unidades, cada una con diferentes habilidades. En otros juegos las unidades pueden evolucionar durante el juego, como sucede en los juegos de estrategia en tiempo real (*RTS*).

Es importante considerar si las unidades, una vez perdidas, no hay forma de recuperarlas, o pueden generarse nuevas del mismo tipo durante el juego. Cuando es posible generar más unidades usualmente se asocia un costo por cada una. Es necesario considerar el costo y cómo balancear la mecánica del juego. Esto se puede lograr dentro del proceso iterativo a través de *playtesting* para determinar los valores adecuados que mantengan el balance del juego, sin dar ventaja a un jugador en particular.

Puntos de vida

Este recurso puede manejarse como atributo de cada objeto en el juego. Al incluir este recurso como parte de la mecánica del juego, permite agregar tensión cuando los puntos de vida son cercanos a cero.

[8]Género de los videojuegos donde se requiere que el jugador utilice su velocidad, destreza y tiempo de reacción para completar un objetivo.

FIGURA A.11: *WarCraft 3*: Diferentes unidades que pertenecen a un jugador (Figura tomada de https://www.projectnerd.it/2015-11-diablo-starcraft-e-warcraft-potrebbero-tornare).

Como parte de la mecánica del juego, es necesario incluir una forma en la que el jugador pueda recuperar puntos de vida, generalmente en juegos de *acción* se colocan objetos para curar al personaje, mientras que juegos *RPG*[9] requieren que se utilice un objeto como comida o una acción de descanso para recuperar puntos de vida. Sin embargo, en juegos de combate (figura A.12) el objetivo principal es acabar con los puntos de vida del adversario antes de perder los propios.

FIGURA A.12: *Mortal Kombat 9*: Barra de puntos de vida de cada jugador (Figura tomada de http://getyourimage.club/resize-17-august.html).

Economía

Al igual que en el mundo real, este recurso facilita el conseguir objetos sin tener que recurrir al trueque. Para esto se debe tener un sistema de economía así como haber definido los procedimientos para realizar los cambios y las reglas que determinen la formas en que se obtiene dinero.

[9] *Role-play Video Game*, es un género donde el jugador controla las acciones de uno o varios personajes dentro de un mundo.

Acciones

En algunos juegos, acciones como turnos o movimientos, son consideradas recursos, por lo que es necesario administrar las elecciones tomadas en cada turno.

Mejoras

Estos recursos usualmente son escasos y difíciles de conseguir, sin embargo otorgan al jugador alguna ventaja dentro del juego, ya sea permanente o temporal, para completar un objetivo.

Inventario

Algunos sistemas permiten al jugador coleccionar y administrar los recursos obtenidos en el juego, algunos sirven para completar objetivos, mientras que otros son de gran valor o muy escasos y posteriormente sirven para cambiarlos por otros objetos o dinero dentro del juego.

Cabe mencionar que en algunos casos, la capacidad del inventario es limitada, por lo que el jugador tendrá que tomar elecciones para administrar sus recursos, ya sea desecharlos, venderlos, o si el juego lo permite, guardarlos en un banco, el cual tiene más capacidad pero no es posible utilizar dichos recursos instantáneamente.

FIGURA A.13: *Resident Evil 4*: Objetos guardados en el inventario, con capacidad limitada (Figura tomada de https://steamcommunity.com/sharedfiles/filedetails/?l=polish&id=318348160).

Tiempo

Este recurso tiene como objetivo lograr que el jugador deba administrar las acciones que realizará teniendo en cuenta el tiempo que toman en completarse. También crea tensión en los últimos instantes, antes de que el tiempo sea agote, en los juegos digitales creando aún más tensión al agregar efectos de sonido.

A.3.6 Conflicto

El conflicto surge cuando el jugador intenta completar los objetivos dentro de las reglas y límites definidos. Lleva al jugador, mediante los procedimientos definidos dentro del juego y el uso de sus habilidades a buscar una manera de completar la tarea asignada[4].

A continuación se listan tres fuentes de conflicto:

- **Obstáculos** - Sirven para dificultar el avance, limitar las habilidades, tener un tiempo mínimo para llevar a cabo acciones (figura A.14). Requieren que el jugador utilice un cierto nivel de habilidades o dedicación para lograr avanzar en el juego.
- **Oponentes** - En este caso el conflicto surge debido a la interacción con otros jugadores (ver patrones de interacción A.3.1).
- **Dilemas** - Las elecciones del jugador tendrán efecto en su experiencia de juego, habilidades e historias diferentes. Estas también dependen de la premisa del juego, dando contexto a las elecciones del jugador.

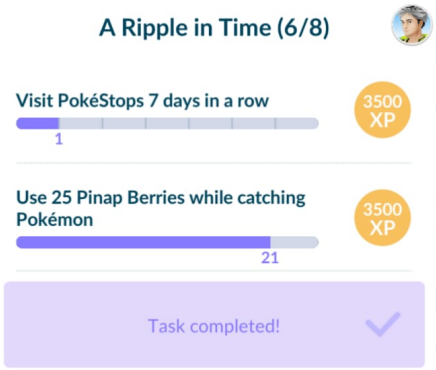


FIGURA A.14: *Pokemón Go*: Obstáculo que depende del tiempo.

A.3.7 Limites

Los límites tienen la función de separar el juego de todo lo demás, como se mencionó anteriormente (sub tema A.3.1), el jugador debe de aceptar voluntariamente las reglas y restricciones para poder considerarse que esta jugando. Es importante el mantener estos límites, ya que si se rompen, sería un juego totalmente diferente, dificultando el lograr obtener la experiencia de juego deseada[4].

Un aspecto a resaltar es que estos límites permiten separar todo lo que sucede en el juego de la vida diaria. Como forma de entretenimiento, los juegos permiten realizar acciones que usualmente no se pueden llevar a cabo en la vida real, sin tener las consecuencias reales al finalizar el juego.

Gracias a los avances de la tecnología, ha sido posible jugar con el alcance de estos límites, permitiendo interactuar con elementos fuera del juego. Por ejemplo, los juegos del género realidad aumentada (*ARGs*), combinan el mundo real con interacciones dentro del juego para crear la mecánica del juego.

Un ejemplo es el videojuego *Pokemon Go*, en el cual el sistema se integra con el *GPS*, Internet y mapas[10], en el cual se debe ir físicamente a los elementos del juego (gimnasios o *PokeStops*) para poder interactuar. Así mismo también ofrece la opción de utilizar realidad aumentada para colocar un personaje del juego sobre lo que la cámara del dispositivo observa e interactuar con el mismo (figura A.15).

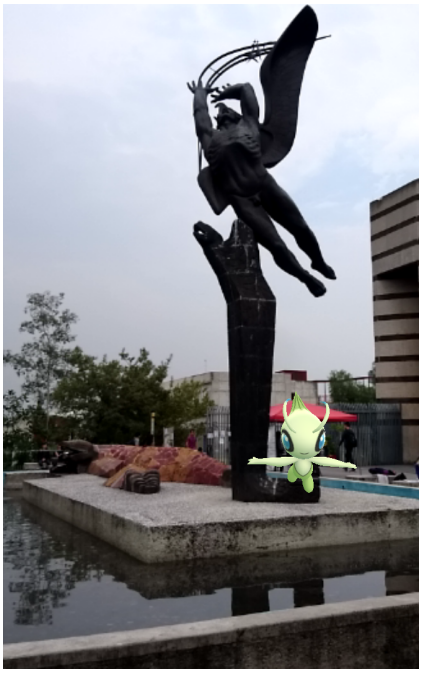


FIGURA A.15: *Pokemón Go*: Realidad aumentada para capturar un personaje.

A.3.8 Resultados

El resultado debe permanecer incierto durante la partida para mantener la atención de los jugadores, ya que si este valor se puede saber con anticipación no sería necesario continuarla. Generalmente se determina de manera imparcial y permite comparar el desempeño de los jugadores, sin embargo no siempre es necesario, ya que existen juegos en los cuales no se define el concepto de ganador o un final, en los cuales existen recompensas de acuerdo a los avances y logros obtenidos[4].

[10]Mapas generados por *OpenStreetMaps*, https://www.openstreetmap.org

Cada juego tiene diferentes formas de calcular el resultado final, sin embargo, siempre depende de los patrones de interacción de los jugadores y el desenlace que tengan (ver patrones de interacción A.3.1) y el objetivo.

A.4 Elementos dramáticos

En la sección anterior se menciona cómo los elementos formales trabajan de manera conjunta para crear un juego, sin embargo, existen otros elementos que permiten involucrar al jugador emocionalmente con la experiencia de juego y enfocarse en la búsqueda de obtener un resultado, estos son los elementos dramáticos. Estos dan contexto al juego, traslapando e integrando los elementos formales del sistema para dar significado al resultado[4].

Los elementos dramáticos básicos como el reto y juego deben estar presentes en todos los juegos. Técnicas dramáticas más complejas como premisa, personaje e historia son utilizadas para explicar y ampliar los elementos abstractos del sistema formal, creando una sensación de conexión para los jugadores con el juego.

Con el fin de crear juegos más atractivos para la población objetivo es importante conocer las relaciones entre estos elementos y cómo trabajan en conjunto con los formales.

A.4.1 Reto

La forma en la que los jugadores ven los retos en un juego depende de las habilidades de cada uno. Sin embargo, es importante tener en cuenta el aprendizaje durante el juego al completar objetivos, al inicio pueden parecer que tienen una alta dificultad, pero al avanzar su nivel de habilitad aumentará, por lo que el diseño del juego debe de contemplar esto e ir regulando la dificultad de los niveles durante todo el juego para mantener el interés del jugador de principio a fin.

Si el reto esta fuera de las capacidades del jugador podría generar aburrimiento, si es fácil, o ansiedad si éste es alto. Llevar a un jugador a estos estados provocaría que dejara el juego.

A.4.2 Juego

El potencial del juego es un elemento clave de los elementos dramáticos que involucra a los jugadores emocionalmente con el juego. Jugar puede definirse como la libertad o movimiento dentro de una estructura rígida. En el caso de los juegos, las restricciones generadas por las reglas y procedimientos son la estructura rígida, y el jugar dentro de dicha estructura es la libertad del jugador para actuar[4].

Para lograr un juego atractivo es necesario identificar los tipos de jugadores que existen. Esta categoría depende de las necesidades y tiempos que las personas dedican para jugar.

- **Competidor**: Juega para ser el mejor, independientemente del juego.
- **Explorador**: Curioso por el mundo, le gusta la aventura, y busca llegar fuera de los límites.
- **Coleccionista**: Busca conseguir todos los objetos, trofeos o conocimiento.
- **Ganador**: Busca completar todos los objetivos del juego, ya sean primarios o secundarios.
- **Bromista**: Solo juega por diversión y no toma el juego seriamente. Algunas ocasiones llega a molestar a los jugadores serios, sin embargo facilita que el juego sea más social que de competencia.
- **Artista**: Prefiere juegos donde se permita más libertad de expresión.
- **Contador de historias**: Se enfoca en seguir la historia del juego.
- **Constructor**: Busca en el juego poder construir objetos.

Esta lista incluye algunos de los tipos de jugadores, cabe mencionar que durante el diseño del juego se debe tomar en cuenta la población a la cual va dirigido para lograr que el juego sea más atractivo.

A.4.3 Premisa

La premisa permite dar contexto al juego, facilita que el jugador pueda hacer elecciones para lograr completar los objetivos y buscar obtener un resultado positivo. Usualmente la premisa se establece como parte de la historia del juego, lo coloca en un tiempo y lugar, con personajes y un contexto que muestra la existencia de un conflicto en el mundo definido por los elementos formales[4].

El principal objetivo de la premisa es permitir que los elementos formales sean atractivos y menos abstractos para el jugador. Más allá de esto, intenta crear un sistema que permita al jugador identificarse con los elementos formales.

Gracias a los avances tecnológicos, los diseñadores han podido incluir premisas más elaboradas en los juegos digitales al punto que pueden ser considerados como historias completas[4].

A.4.4 Personaje

Los personajes son los elementos a través de los cuales el jugador puede tener interacción con los elementos dentro del juego utilizando los procedimientos definidos para avanzar en la historia y completar objetivos[4].

El personaje principal de la historia, también conocido como protagonista, es quién actúa para resolver el conflicto a través de la historia, cuyas elecciones pueden tener impacto en el desarrollo de la misma. El rol de los personajes se define dentro de la historia, su función, lo que dicen, las acciones que toman y sus relaciones con otros personajes.

Sin importar la complejidad del personaje y su rol dentro del juego, es importante tener en cuenta durante el diseño cuatro puntos esenciales para asegurar que el personaje tiene presencia en la historia:

- ¿Qué es lo que busca?
- ¿Qué es lo que necesita?
- ¿Qué es lo que el jugador espera?
- ¿Qué es lo que el jugador teme?

Algunas herramientas que permiten al jugador identificarse con el personaje son:

- **Caracterización**: Es la forma en la que luce, la importancia dada por el contexto y la premisa, así como los valores que representa.
- **Desarrollo del personaje durante la historia**: Son los cambios que va asumiendo durante la historia, ya sean de apariencia, habilidades, la forma en que se relaciona con otros personajes o su caracterización.
- **Contexto**: Es el lugar en donde se desarrolla la historia, cuáles son los conflictos que existen y cuál es la importancia del personaje dentro de ésto.
- **Motivación**: Cuál es la recompensa que tiene el personaje al completar los objetivos y resolver el conflicto dentro del contexto definido.
- **Creación del personaje**: Permite al jugador crear el personaje incluyendo rasgos con los que se puede identificar, apariencia, habilidades, contexto y metas.

A.4.5 Historia

El resultado de la historia debe permanecer incierto (al menos durante la primera vez que se juega). Juegos, películas, libros, series de televisión o cualquier forma de entretenimiento que narre una historia debe iniciar incierta y resolverse con el avance de la misma. Es importante mencionar que en los videojuegos el resultado se va dando por las acciones de los jugadores, a diferencia de otros medios en donde el autor es quien va guiando los avances[4].

En algunos juegos la historia es rígida, ofrece contexto, premisa, conflicto y motivación de los personajes, sin embargo el progreso no es afectado por las elecciones de los jugadores.

Por otra parte, para ofrecer una mayor sensación de control a los jugadores, algunos videojuegos permiten que las acciones del jugador modifiquen el curso de la historia, así como el resultado final. Existen dos estructuras en las que las elecciones de los jugadores afectan el curso de la historia:

- **Ramificaciones**: El jugador elige entre varias posibilidades en algunos puntos de la historia causando predeterminados cambios. Estas opciones están restringidas por la estructura de la historia, provocando que en algunos puntos el juego carezca de reto, y tenga resultados poco interesantes.
- **Emergente**: Se deja al jugador la libertad de interactuar con los objetos definidos en los elementos formales, y la historia se va formando como efecto de la interacción entre los objetos utilizando los procedimientos y manteniéndose dentro de las reglas.

A.4.6 Creación del mundo

Como complemento al diseño de la historia, se debe crear el universo en donde se desarrolla. Este mundo ficticio esta compuesto inicialmente por mapas e historia, sin embargo puede llegar a contener todos los elementos de una civilización como cultura, economía, lenguajes, política, razas, etc. Una gran variedad de películas y juegos generan mundos ficticios para ofrecer más detalles a la historia y mantener a los jugadores interesados por más tiempo. Un ejemplo de esto es el universo de *Star Wars* en el que se han desarrollado películas, series de televisión, videojuegos, libros y una gran cantidad de material creado por el público[4].

Para crear un mundo ficticio es importante integrar la mayor cantidad de propiedades de medios de comunicación. Se le llama *diseño inmersivo* cuando se utilizan diferentes plataformas para crear mundos que sean coherentes, tengan lógica, historia, geografía, superficie y reglas. Este proceso de diseño involucra producción digital y virtual que permite la colaboración de una amplia variedad de disciplinas en todos los aspectos del mundo. Para crear un mundo se unen los esfuerzos de diseñadores de: películas, moda, juegos, teatro, televisión, música, arquitectura, ciencia, medios interactivos, entre otros[4].

A.4.7 Arco dramático

El conflicto es el núcleo del drama, que funciona en conjunto con los elementos formales del juego, el cual está diseñado para evitar que el jugador cumpla los objetivos fácilmente, así como atraer a los jugadores emocionalmente creando una sensación de tensión antes de obtener un resultado. El manejo de esta tensión es importante para el éxito de un juego, así como una novela o película[4].

En el drama, el conflicto surge cuando el protagonista enfrenta un problema u obstáculo que evita que consiga completar el objetivo. Una vez iniciado el conflicto, debe ir aumentando para que el drama sea efectivo, es decir, aumentar la tensión, la historia debe empeorar antes de mejorar, lo resultante es un arco dramático.

Como se puede observar en la figura A.16, la historia comienza con la *exposición*, en la cual se introducen los personajes, el contexto y los conceptos que serán importantes. El conflicto surge cuando el protagonista tiene una meta y existen obstáculos. El *aumento de la acción* inicia cuando el protagonista lleva acabo acciones para resolver el conflicto. Posteriormente a esto se llega al *climax*, donde se integra un factor decisivo o un evento importante, lo cual determinará el resultado del drama. Después de esto, inicia la *caída de la acción* donde el conflicto comienza a resolverse y al final del arco se encuentra la *resolución*, en donde queda resuelto el conflicto.

En un videojuego, el *aumento de la acción* depende de los elementos formales y dramáticos, ya que están diseñados para aumentar la dificultad del reto conforme progresan los avances y se desarrolla la historia.

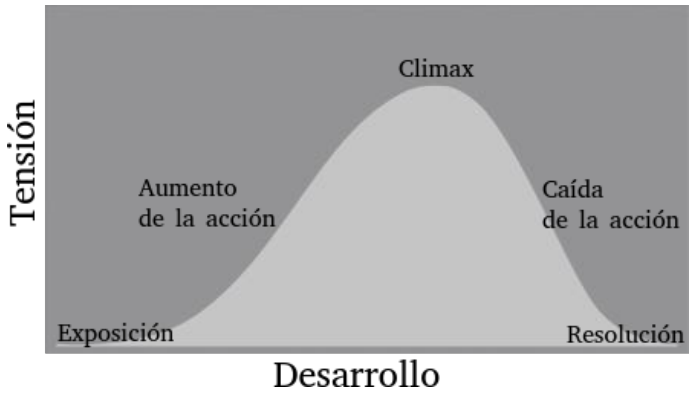


FIGURA A.16: *Arco dramático*:Estructura de todos los medios que incluyen drama, incluidos los videojuegos (Figura adaptada de [4]).

Los elementos dramáticos mencionados en este capítulo, son herramientas importantes para transmitir sentimientos y emociones al jugador. El impacto emocional de los videojuegos ha crecido hasta estar a la par de películas, sin embargo no lo suficiente para ser reconocido como una forma de arte dramática. En la actualidad los diseñadores buscan formas más sofisticadas para integrar narrativa, contexto, profundidad de los personajes y la intención dramática en general[4].

A.5 Sistemas del juego

Un sistema se define como un conjunto de elementos que interactúan entre ellos con una meta en común, éstos se presentan tanto en el mundo natural como en lo creado por los seres humanos,

en los cuales se puede observar un comportamiento emergente a través de la interacción entre los elementos.[4]

A.5.1 Juegos como sistemas

En el caso de los juegos, que están compuestos por un conjunto de elementos formales, al unirse y funcionar para completar una meta, crean una experiencia de juego dinámica para cada jugador. Estos sistemas pueden pueden producir resultados precisos y predecibles o una gran variedad de efectos impredecibles, elegir qué tipo de sistema se adapta al juego durante la fase de diseño es importante ya que los resultados que puede ofrecer dependen de esto, un juego en el que existan una o dos posibilidades de resultados o un sistema impredecible en donde los resultados se definan por las elecciones de los jugadores y las interacciones del jugador con los elementos formales[4].

La forma en la que funciona un sistema depende de los elementos básicos que lo conforman y las interacciones que existen entre éstos, así como el resultado que se obtiene derivado de dichas interacciones.

Los elementos básicos del sistema son objetos, propiedades, comportamientos y relaciones. Los objetos dentro del sistema interactúan con otros de acuerdo a sus propiedades, comportamientos y relaciones, generando cambios en el estado del sistema.

- **Objetos**: Son los elementos básicos que constituyen un sistema, físicos o abstractos, los cuales tienen propiedades y comportamientos definidos por las reglas.
- **Propiedades**: Cualidades o atributos que definen aspectos de cada objeto.
- **Comportamientos**: Potenciales acciones de un objeto en un estado del sistema.
- **Relaciones**: La forma en la que interactúan los objetos entre si, dependiendo de las propiedades y el estado en que se encuentran dentro del juego.

A.5.2 Sistemas dinámicos

Un sistema, por definición requiere que todos los elementos estén presentes y trabajen de manera conjunta para lograr una meta. Deben estar colocados de manera precisa con un propósito para proveer el reto adecuado, si algo es modificado los resultados de la interacción pueden cambiar dependiendo de las interacciones entre los elementos, por lo que habrá un cambio, ya sea poco notable o que impacte en la mecánica del juego[4].

Una propiedad importante de los sistemas es que la suma de sus elementos es mayor, por lo que poner atención a cada elemento no describe como se comportará el juego.

Sistemas emergentes

Los sistemas pueden producir resultados impredecibles, sin embargo no es necesario que los sistemas sean complejos. Basta con tener un conjunto de reglas simples para tener resultados complejos, a esto se le llama *sistemas emergentes*.

Un ejemplo de esto es el sistema de *Conway*, en el que se definen tres reglas sobre una cuadrícula, en la que se tienen dos estados por casilla, prendido y apagado, vivo y muerto respectivamente. En el cual se encontró que diferentes estados iniciales podían evolucionar, a través de iteraciones, en una gran variedad de estados, incluso tener agentes que *caminan* (figura A.17) o compuertas lógicas[4].

Utilizando solo las ocho celdas adyacentes a la que se va a evaluar:

- **Nacer**: si está apagada y exactamente tres adyacentes están prendidas, se prende.
- **Muerte por soledad**: si está prendida y menos de dos adyacentes están prendidas, se apaga.
- **Muerte por sobre población**: si está prendida y al menos cuatro adyacentes están prendidas, se apaga.

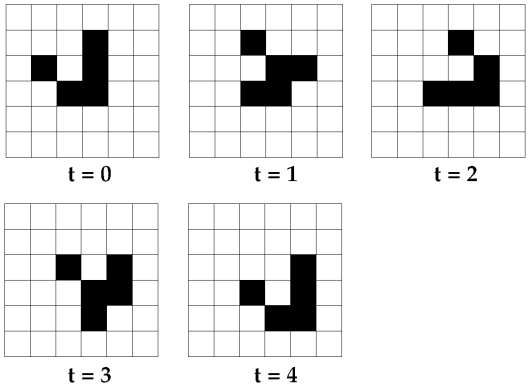


FIGURA A.17: *Glider*: configuración para generar un agente con la habilidad de caminar (Figura adaptada de https://en.wikipedia.org/wiki/Conway's_Game_of_Life).

A.5.3 Interacción con el sistema

Entre los elementos formales se encuentran los procedimientos, a través de los cuales los jugadores pueden interactuar con los elementos del juego. Los aspectos que hay que considerar para que el jugador pueda tener control sobre las acciones que llevará a cabo son[4]:

- La información que tiene del estado del sistema.
- Los aspectos que puede controlar del sistema.

- Cómo está estructurado el sistema.
- La retroalimentación que recibe el jugador por parte del sistema.
- La forma en que afecta a la mecánica del juego.

Estructura de la información

Para que los jugadores puedan realizar una elección de sus acciones en el juego, es importante que conozcan información del estado de los objetos del juego y las relaciones que existen entre estos. Esto influye en la sensación del control por parte del jugador.

La cantidad de información que recibe el jugador acerca de los objetos dentro del juego puede cambiar durante el transcurso del juego, lo cual también puede modificar la dificultad reduciéndola, como parte de la mecánica del juego.

Control

Los controles de un juego están directamente relacionados a su diseño físico, en juegos de mesa o cartas ofrecen la posibilidad de manipular directamente los elementos, por otra parte los juegos digitales utilizan algún dispositivo de entrada como teclado, ratón o controles. Estos permiten modificar el estado del juego de manera directa. Algunos otros juegos permiten al jugador modificar variables para alterar el estado del sistema de manera indirecta.

Retroalimentación

Implica una relación directa entre los datos de entrada para una interacción y un cambio en otro elemento del sistema, esto puede ser positivo o negativo. En el caso de la retroalimentación positiva, se recompensa al jugador para reforzar un efecto positivo, por otra parte, en la retroalimentación negativa permite balancear el sistema tratando de compensar a los otros jugadores o el sistema[4].

A.5.4 Ajustes de los sistemas del juego

Se debe asegurar que el sistema es internamente completo, es decir, que las reglas definidas no lleven a algún ciclo sin salida durante el juego, que permita que el jugador avance sin resolver el conflicto o que no pueda resolverse[4].

También deben de realizarse pruebas para asegurar que el sistema está balanceado, no dar ventaja a un jugador y dar oportunidades por igual para cumplir los objetivos, de lo contrario perderán el interés y abandonarán el juego por sentir que el sistema hizo trampa.

Finalmente, se deben realizar pruebas para asegurar que se obtiene la experiencia de juego deseada con un nivel de reto adecuado para la población a la que va dirigido el juego, balanceando la dificultad del sistema.

Uno de los retos principales en el diseño y ajuste del sistema de juego es separar los objetos o relaciones que generan problemas durante la experiencia de juego y repararlos sin generar nuevos. Cuando los elementos del sistema trabajan de manera conjunta surge una buena mecánica de juego[4].

Apéndice B

Cuestionario

1. Edad: _____ años
2. ¿Te gustan los videojuegos?
   [ ]Si [ ]No
3. ¿Cuántas horas dedicas por semana a jugar videojuegos?
   _____ horas
4. ¿En qué plataformas prefieres jugar?
   [ ]PC [ ]Play Station [ ]XBox [ ]Switch [ ]Móvil
   Otros:_____
5. ¿Porqué prefieres esa plataforma?_____
   _____
6. ¿Te gustan los videojuegos del género de plataformas?
   [ ]Si [ ]No
7. En caso de que si te gusten:
   - Menciona algunos videojuegos de este género que hayas jugado:_____
     _____
   - ¿Qué es lo que más te gusta?_____
     _____

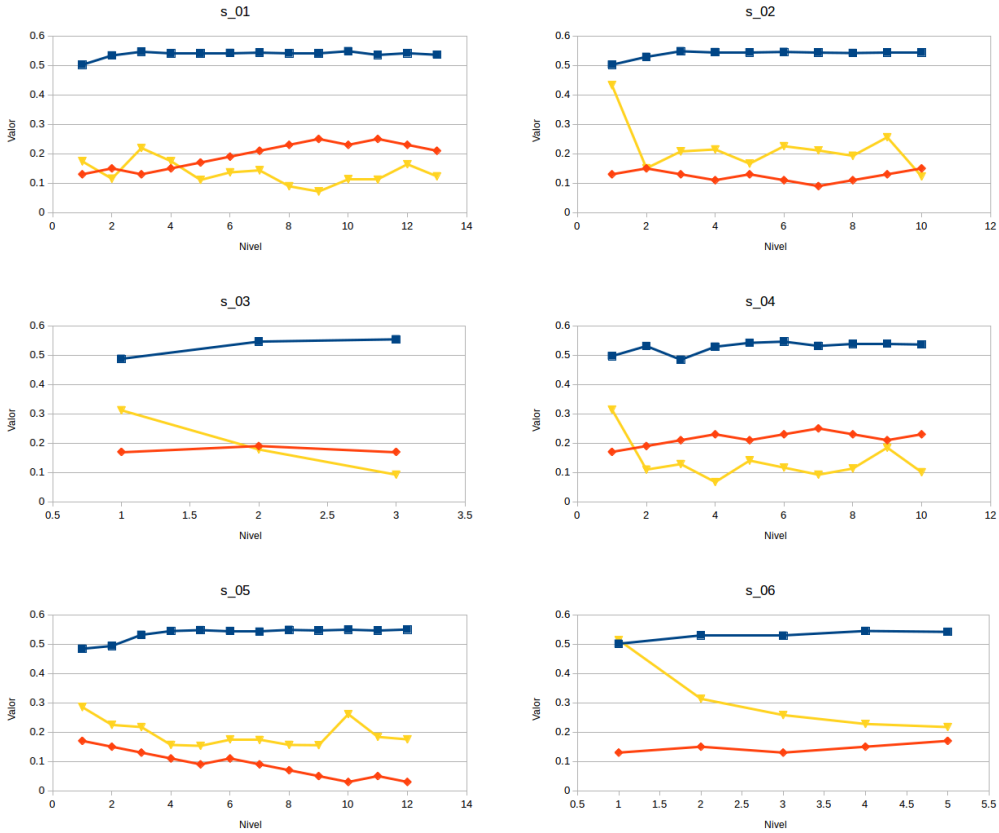
   _____
8. ¿Pudiste completar todos los niveles?
   [ ]Si [ ]No
   - En caso de que no, ¿Porqué?_____
     _____

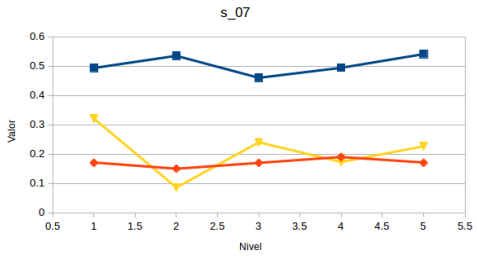
9. ¿Qué te pareció el diseño de los niveles?
   [ ]Muy bueno [ ]Bueno [ ]Normal [ ]Malo [ ]Muy malo
10. ¿Qué te pareció la longitud de los niveles?
    [ ]Muy largo [ ]Largo [ ]Normal [ ]Corto [ ]Muy corto
11. ¿Qué te pareció el contenido de los niveles?
    [ ]Muy bueno [ ]Bueno [ ]Normal [ ]Malo [ ]Muy malo
12. En promedio, ¿Cuál fue la dificultad de los niveles?
    [ ]Muy difícil [ ]Difícil [ ]Normal [ ]Fácil [ ]Muy fácil
13. ¿Hubo alguno que fuera demasiado fácil?
    [ ]Si [ ]No
    - En caso de que si, ¿Porqué?_____
14. ¿Hubo alguno que fuera demasiado difícil?
    [ ]Si [ ]No
    - En caso de que si, ¿Porqué?_____
15. ¿Notaste algún patrón entre las acciones tomadas durante el nivel con respecto al siguiente?
    [ ]Si [ ]No
    - En caso de que si, explica brevemente:_____

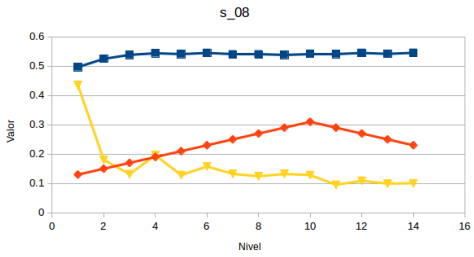
Apéndice C

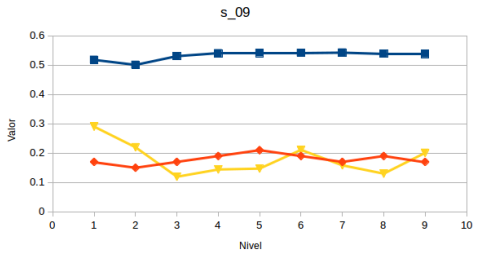
Gráficas de resultados

En esta sección se muestran las gráficas de valores por persona que realizó la prueba. En el eje horizontal se muestran los niveles generados, mientras que en el eje vertical, en color azul se muestra la dificultad, en rojo la linealidad y en amarillo el tiempo que tardó en generarse cada nivel.




s_07
Valor
Nivel
0.6
0.5
0.4
0.3
0.2
0.1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5


s_08
Valor
Nivel
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10
12
14
16


s_09
Valor
Nivel
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5
6
7
8
9
10

Bibliografía

[1] Jenova Chen. “Flow in Games”. MA thesis. URL: http://jenovachen.com/flowingames.

[2] Jenova Chen. “Flow in Games (and Everything Else)”. In: *Communications of The ACM* 50.4 (2007).

[3] Kate Compton y Michael Mateas. “Procedural Level Design for Platform Games”. In: *American Association for Artificial Intelligence* (2006).

[4] Tracy Fullerton. *Game Design Workshop*. 3rd. CRC Press, 2014.

[5] Héctor Adrián Díaz Furlong. “Generación Procedural de Contenido en la Programación de Videojuegos”. Maestría en Ciencias. U.N.A.M., 2013.

[6] Elisa Viso Gurovich. *Introducción a la teoría de la computación(utómatas y leguajes formales)*. Facultad de Ciencias, 2008.

[7] Sergey Karakovskiy y Julian Togelius. “The Mario AI Benchmark and Competitions”. In: *IEEE Transactions on Computational Intelligence and AI in Games* (2012). URL: http://julian.togelius.com.

[8] Ahmed Khalifa y Julian Togelius. “Marahel: A Language for Constructive Level Generation”. In: *Association for the Advancement of Artificia Intelligence* (2017). URL: http://julian.togelius.com.

[9] A. Lagae, S. Lefebvre y T. DeRose. “A Survey of Procedural Noise Functions”. In: *Eurographics Association and Blackwell Publishing Ltd.* (2010).

[10] Melanie Mitchell. *An Introdution To Genetic Algorithms*. MIT Press Cambridge, MA, USA, 1996.

[11] Christoffer Pedersen, Julian Togelius y Georgios Yannakakis. “Modeling Player Experience for Content Creation”. In: *IEEE Transactions on Computational Intelligence and AI in Games* (2010). URL: http://julian.togelius.com.

[12] Noor Shaker y Julian Togelius. “The 2010 Mario AI Championship: Level Generation Track”. In: *IEEE Transactions on Computational Intelligence and Games* (2010). URL: http://julian.togelius.com.

[13] Noor Shaker, Julian Togelius y Mark J. Nelson. *Procedural Content Generation in Games*. Springer, 2016.

[14] Noor Shaker, Georgios N. Yannakakis y Julian Togelius. “Evolving Levels for Super Mario Bros Using Grammatical Evolution”. In: *IEEE Conference on Computational Intelligence and Games* (2012). URL: http://julian.togelius.com.

[15] Tanya X. Short y Tarn Adams. *Procedural Generation in Game Design*. CRC Press, 2017.

[16] Gillian Smith y Jim Whitehead. “Analyzing the Expressive Range of a Level Generator”. In: *Proceedings of the 2010 Workshop on Procedural Content Generation in Games* (2010).

[17] Gillian Smith et al. “Launchpad: A Rhythm-Based Level Generator for 2-D Platformers”. In: *IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES* 3.1 (2011).

[18] Penelope Sweetser y Peta Wyeth. “GameFlow: A Model for Evaluating Player Enjoyment in Games”. In: *ACM Computers in Entertainment* 3.3 (2005).

[19] Julian Togelius et al. “The Mario AI Championship 2009-2012”. In: *AI Magazine* 34.3 (2013), pp. 83–92. URL: http://julian.togelius.com.

[20] Julian Togelius et al. “What is procedural content generation? Mario on the borderline”. In: *Proceedings of the FDG Workshop on Procedural Content Generation* (2011). URL: http://julian.togelius.com.