VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

Generación procedural de contenidos en entornos computacionales y artísticos: una experiencia inmersiva

Procedural Content Generation in Computational and Artistic Environments: an immersive experience

Alan Ricardo Hernández Pérez[1], Fernando Salvador Barrón Zaragoza[1], Mirhan Amely Ponce Jiménez[1], Rocío Alfonsina Lizárraga Morales[2], Uriel Haile Hernández Belmonte[2]

[1] Licenciatura en Artes Digitales, División de Ingenierías, Campus Irapuato-Salamanca, Universidad de Guanajuato.
[2] Departamento de Arte y Empresa, División de Ingenierías, Campus Irapuato-Salamanca, Universidad de Guanajuato.
ar.hernandezperez@ugto.mx, fs.barronzaragoza@ugto.mx, ma.poncejimenez@ugto.mx, ra.lizarragamorales@ugto.mx, uh.hernandez@ugto.mx

Resumen

La generación procedural de contenido (PCG) es una forma automatizada de creación que se basa en algoritmos de programación. La generación procedural está basada en parámetros que pueden ser pseudoaleatorios o definidos por el programador con información limitada, nula o indirecta. La PCG en los entornos artísticos facilita procesos que, al realizarse manualmente, pueden resultar de alto costo o ineficientes. En este proyecto se explora la implementación de la PCG para crear entornos tridimensionales y síntesis de música en tiempo real para una experiencia inmersiva de Realidad Virtual. Esto se logró realizar a través de nodos de geometría de Blender en donde fueron implementados algoritmos como los diagramas de Voronoi. Así mismo, la síntesis de música se realizó con el software de CSound como cerebro que controla las notas musicales agregadas en Cadenas de Markov, que posteriormente se envía a un DAW para ser procesadas y ejecutadas. Como resultado, se creó una experiencia inmersiva en Realidad Virtual titulada “En el umbral de un corazón nostálgico”, una ciudad procedural que consta de edificios, islotes, cuerpos de agua, escaleras, umbrales y ambientación musical, con parámetros que modifican su composición para crear una experiencia única en cada ejecución.

Palabras clave: Generación Procedural, Algoritmos, Realidad Virtual, 3D, diagramas de Voronoi, Cadenas de Markov, Nodos de Geometría, Ruido de Perlin.

Introducción

La generación procedural de contenido (PCG, por sus siglas en inglés) es una forma de creación automática de contenido mediante el uso de algoritmos en lugar de crearlo de forma manual. Este contenido es generado a partir de parámetros iniciales con entrada de información nula, limitada o indirecta de los programadores y artistas. Utilizando procesos pseudoaleatorios o aleatorios, la PCG es capaz de generar contenido en un rango a priori, imposible de predecir (Sánchez & Romero, 2018/2019). Existen diferentes tipos de algoritmos de generación procedural como los basados en autómatas celulares, en grafos, en reglas y otros más avanzados como los basados en inteligencia artificial. Entre las aplicaciones más populares se encuentran las diferentes áreas de las artes digitales como: gráficos por computadora, audio digital, animación, ilustración, entre otras (Orrego, 2019).

Se ha creído erróneamente que la base de la PCG es completamente aleatoria, cuando en realidad el contenido se genera a partir de rangos establecidos en parámetros iniciales definidos por el programador dentro del entorno desarrollado y diseñado artísticamente. Esta forma de creación cumple múltiples funciones en el ámbito artístico. La aplicación de técnicas de PCG ha cobrado relevancia, ya sea en videojuegos o en experiencias inmersivas. Esto se debe a que la creación manual de entornos únicos o de alta densidad, como bosques o ciudades, es costosa e ineficiente, por lo que su generación mediante algoritmos con parámetros manipulables constituye una alternativa viable y de bajo costo para los artistas.

La PCG necesita elementos de diseño, no solo algorítmicos, sino también artísticos. Algunos de los casos en los que la complejidad del resultado visual no solo abarca lo técnico, sino también lo artístico o de diseño son: la generación procedural de música, en la que también hay que tener en cuenta el ritmo, las escalas musicales

o incluso los silencios. En la creación procedural de vegetación o terreno, se debe tener en cuenta la variación natural que existe y la coherencia de tamaños sin cambios abruptos. En la PCG de modelos 3D se debe tener en cuenta la topología o la cantidad de polígonos para un buen rendimiento en el renderizado (Hamdani, 2026). Algunos de los videojuegos que utilizan la generación para crear entornos son No Man's Sky de 2016 (Hello Games, 2016), que genera un escenario interminable a medida que el jugador explora y crea mundos completos con vegetación, fauna, clima y muchos otros parámetros para que cada mundo sea diferente. En el videojuego Minecraft (Studios, Mojang, 2011), los mundos se crean a partir de mapas de elevación, densidad volumétrica y algoritmos como el ruido de Perlin, para determinar el límite de cada bioma generado.

En este proyecto se exploró la creación de un entorno tridimensional con exploración inmersiva en Realidad Virtual (VR), donde tanto el entorno 3D como la ambientación sonora se crean con algoritmos computacionales. A través de la variación de parámetros de dichos algoritmos, se generan variaciones de la experiencia para que ésta sea única en cada ejecución. Se plantea el desarrollo de una experiencia inmersiva desplegada en los visores Meta Quest S3 con un enfoque en la corriente artística del surrealismo. Este proyecto se titula "Umbral de un corazón nostálgico". Esta experiencia audiovisual consta de un entorno tridimensional conformado por edificios, vegetación, cuerpos de agua, así como portales y escaleras flotantes, acompañado de música y sonidos de diseño sonoro, todo generado de manera procedural, posibilitando cambiar parámetros que crean un mundo diferente en cada ejecución. Para la generación procedural se utilizaron distintos algoritmos base como semillas de Voronoi para la ubicación de los edificios, el ruido de Perlin para la generación irregular de terreno o las cadenas de Markov para la síntesis de música en vivo. Integramos los algoritmos para la creación de un entorno en 3D a través de Blender que permite la navegación inmersiva de Realidad Virtual con el Add-on de VR Scene Inspection.

Para que la inmersión de nuestro proyecto sea aún más consistente, creamos una historia que acompaña a la experiencia, para dar contexto al significado artístico de los mundos creados en nuestro proyecto y crear una atmósfera acorde con la música y nuestros escenarios. La historia es la siguiente:

“En el umbral de un corazón nostálgico”

En el abismo entrañable de una ciudad sin seres existentes, aún quedaba alguien que mantenía la esperanza de revivir lo que algún día fue. Solo existe la edificación que mantiene los recuerdos de eso que existió; recorrer los arbustos y las flores le recuerdan lo que aún queda de su alma, y unas piedras le recuerdan sus errores. Todo se ve borroso por la niebla de las preguntas que aún tiene sin responder. Está harto de aquellos portales que en realidad son puertas hacia los malos momentos que desea borrar, pero sigue cruzándolos, esperando que por lo menos la próxima vez pueda cambiar algo y todo sea diferente. A veces pierde la cordura, pero cuando siente que ya no puede más, siempre regresa esa señal que le recuerda que aún hay mucho por qué luchar…

El resto del artículo se organiza de la siguiente manera: en la sección 2 se describe la metodología utilizada para cada elemento integrado en Umbrales de la Memoria, desde la creación del terreno irregular, los cuerpos de agua, la ubicación de los edificios, los modelos de vegetación, la síntesis de música y demás elementos procedurales dentro del proyecto. En la sección 3, se muestran los resultados alcanzados a través de renders de la ciudad y sus variables que hacen única cada experiencia. Por último, en la sección 4, se presentan las conclusiones generales del proyecto.

Metodología

La PCG que implementamos en este trabajo contiene dos variaciones de generación de contenido: creación de entornos tridimensionales y síntesis de música en tiempo real. Ambas aproximaciones se basan en la creación automatizada de elementos mediante algoritmos, reglas matemáticas y parámetros definidos por el artista, en lugar del modelado puramente manual, que posteriormente se mezclan y generan un entorno inmersivo. El entorno inmersivo se puede ejecutar de múltiples maneras ante el espectador, una de ellas a través de la Realidad Virtual o VR con visores adaptados como lo son los Meta Quest S3.

Como se mencionó anteriormente, en este proyecto exploramos la creación de entornos tridimensionales a través de mallas 3D (van Gumster, 2022). Una malla 3D no es un objeto sólido, sino que son datos geométricos suspendidos en un espacio virtual. Las mallas 3D están compuestas por 3 elementos fundamentales: 1) Vértices, que son puntos creados a partir de coordenadas cartesianas calculadas matemáticamente, 2) Edges o bordes, que son las aristas que conectan los vértices entre sí, y 3) Faces o caras, que son los polígonos que se generan a partir de que 3 bordes o más se cierran. Las faces o caras son la parte que al final se renderiza, texturiza y a la que se le agregan colisiones.

El desarrollo de este proyecto se realiza en un flujo de trabajo generativo implementado dentro del software Blender, centrado específicamente en el uso del sistema paramétrico de Nodos de Geometría. Este enfoque facilita el modelado procedural y la manipulación matemática de atributos en tiempo real, permitiendo controlar dinámicamente la escala, densidad y distribución de las instancias tridimensionales del entorno. A diferencia del modelado poligonal estático, la implementación de los Nodos de Geometría permite la deformación y generación de mallas 3D en tiempo real, a través de programas escritos en el lenguaje de programación Python. Con el propósito de unificar la experiencia sensorial, la lógica geométrica creó dos códigos que operan en paralelo, limitando la movilidad de la cámara de Realidad Virtual, generando la sensación de caminata y haciendo más inmersiva la experiencia.

Como se mencionó anteriormente, esta experiencia audiovisual consta de un entorno tridimensional conformado por edificios, vegetación, cuerpos de agua, portales y escaleras flotantes, acompañado de música y sonidos de diseño sonoro. Para la generación procedural se utilizaron distintos algoritmos base como los Diagramas de Voronoi para la ubicación de los edificios, el ruido de Perlin para la generación irregular de terreno y las cadenas de Markov para la síntesis de música en vivo.

El diagrama de Voronoi es una partición del plano euclidiano en regiones asociadas a un conjunto finito de puntos generadores, también denominados sitios o semillas. Cada región contiene todos los puntos del plano cuya distancia al sitio correspondiente es menor o igual que la distancia a cualquier otro sitio (Aurenhammer, 1991). Dado un número de puntos en el plano, su diagrama de Voronoi divide el plano de acuerdo con la regla del vecino más cercano (Ver Figura 1): cada punto es asociado con la región del plano más cercana a él. Sea un conjunto finito de n sitios generadores, la celda de Voronoi asociada al sitio p_i se define como:

$$V(p_i) = \{x \in \mathbb{R}^2 \mid d(x,p_i) \leq d(x,p_j), \forall j \neq i\}.$$

En este caso, $d(x,p_i)$ representa la distancia euclidiana entre el punto x y el sitio p_i. Por lo tanto, el diagrama de Voronoi correspondiente al conjunto P puede expresarse como:

$$Vor(P) = \{V(p_1), V(p_2), \ldots, V(p_n)\}.$$

Cuando dos celdas son adyacentes, su frontera está formada por los puntos que se encuentran a la misma distancia de los dos sitios generadores correspondientes. En el plano bidimensional, estas fronteras se encuentran sobre las mediatrices de los segmentos que unen dichos sitios y pueden adoptar la forma de segmentos, semirrectas o rectas. De esta manera, las celdas cubren completamente el plano y sus regiones interiores no presentan traslapes. En esta exploración se optó por utilizar los sitios de Voronoi para la ubicación estratégica de los edificios.

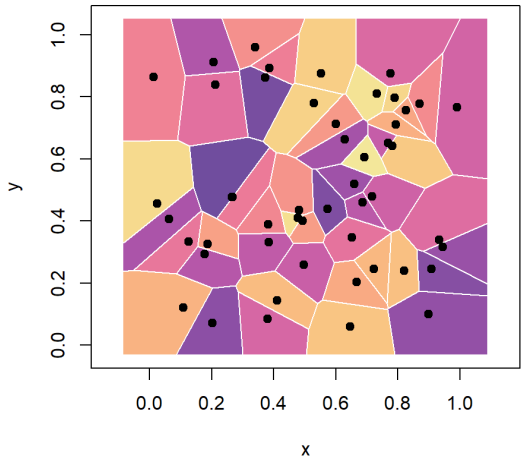


Figura 1. Ejemplo de visualización de los diagramas de Voronoi para un conjunto de puntos en el espacio.

Además de explorar la creación de mallas tridimensionales, se exploró la creación musical a partir del algoritmo de las cadenas de Markov. Las cadenas de Markov funcionan a partir de probabilidades conforme a matrices de transición previamente programadas (Figura 2). Es decir, aquí los datos de creación procedural pueden ser programados para tener más control sobre el sonido a partir de un algoritmo que puede ser pseudoaleatorio. Al adoptar las cadenas de Markov, el sistema calcula el evento sonoro basándose en el estado actual para crear el flujo musical, optimizando el rendimiento computacional durante la síntesis en tiempo real.

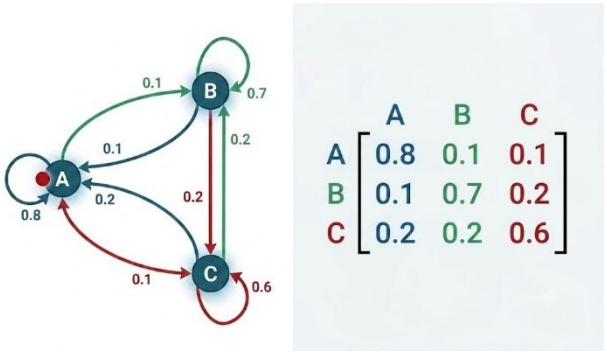


Figura 2. Ejemplo de visualización de las cadenas de Markov.

En las siguientes subsecciones, se mostrará la creación de los elementos incluidos dentro de nuestro mundo procedural inmersivo.

Ciudad.

El sistema de generación de la ciudad se desarrolló mediante un árbol de nodos de geometría en Blender (Lens, 2023). La lógica de construcción parte de una malla base tipo grilla, en la que cada celda funciona como un lote potencial. A partir de esta unidad se derivan los vacíos, los islotes, los cúmulos de tierra, las edificaciones y los elementos secundarios como vegetación, ventanas y escaleras con portales.

La elección de una grilla como punto de partida responde a la necesidad de contar con una estructura controlable, divisible y parametrizable. Mediante los parámetros de tamaño y resolución, el usuario puede modificar la escala total de la ciudad y la cantidad de subdivisiones disponibles para esta. Esto permite que una misma lógica de generación funcione tanto para ciudades pequeñas como para ciudades más densas, sin tener que reconstruir manualmente el escenario. Sobre esta grilla inicial se aplica una primera etapa de selección y eliminación de caras. Esta operación utiliza valores pseudoaleatorios controlados por semillas, combinados con patrones de ruido y una distribución de tipo Voronoi. La pseudoaleatoriedad permite generar variaciones distintas a partir del mismo sistema.

El resultado de esta primera etapa es una máscara de ocupación donde algunas caras permanecen activas y otras son eliminadas. Las caras que no se eliminan se interpretan como islotes, mientras que las caras eliminadas generan vacíos dentro de la composición. A partir de este punto, el árbol de nodos se divide en dos ramas principales: la rama A, dedicada a la generación de cúmulos de tierra y vegetación, y la rama B, dedicada a la generación de edificios y a los elementos asociados a ellos. (ver Figura 3).

VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

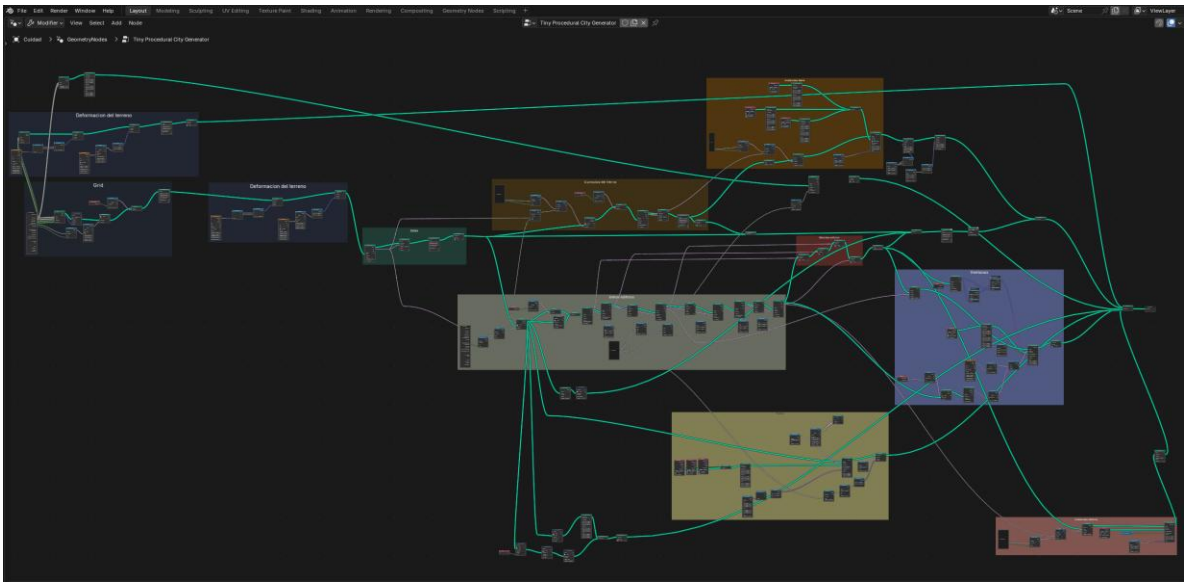


Figura 3. Árbol de nodos que construyen la ciudad.

Rama A: generación de cúmulos de tierra y vegetación

La rama A toma como entrada las caras supervivientes de la grilla y aplica una nueva selección pseudoaleatoria para determinar qué partes y que área de los islotes se convertirán en cúmulos de tierra. Estas zonas se extruyen en sus tres ejes XYZ, generando masas irregulares sobre la grilla.

La finalidad de esta rama no es únicamente crear un relieve, sino también generar una máscara de selección. Las áreas que pasan a formar parte de los cúmulos de tierra se almacenan como zonas válidas para la aparición de vegetación. Con esto, la vegetación no se distribuye de manera libre sobre toda la ciudad, sino únicamente sobre las áreas que cumplen la regla de ser terreno natural.

A partir de dicha máscara, se distribuyen puntos sobre las superficies seleccionadas y se instancian modelos de vegetación. La posición, escala y rotación de cada instancia pueden variar mediante valores pseudoaleatorios para evitar repeticiones visibles. De esta manera, la vegetación se comporta como un sistema procedural condicionado: aparece automáticamente en las zonas correctas y se actualiza en tiempo real si cambia la semilla, la resolución de la grilla o la forma de los islotes. En la Figura 4 se visualizan los nodos que generan la tierra y la vegetación de nuestro entorno.

VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

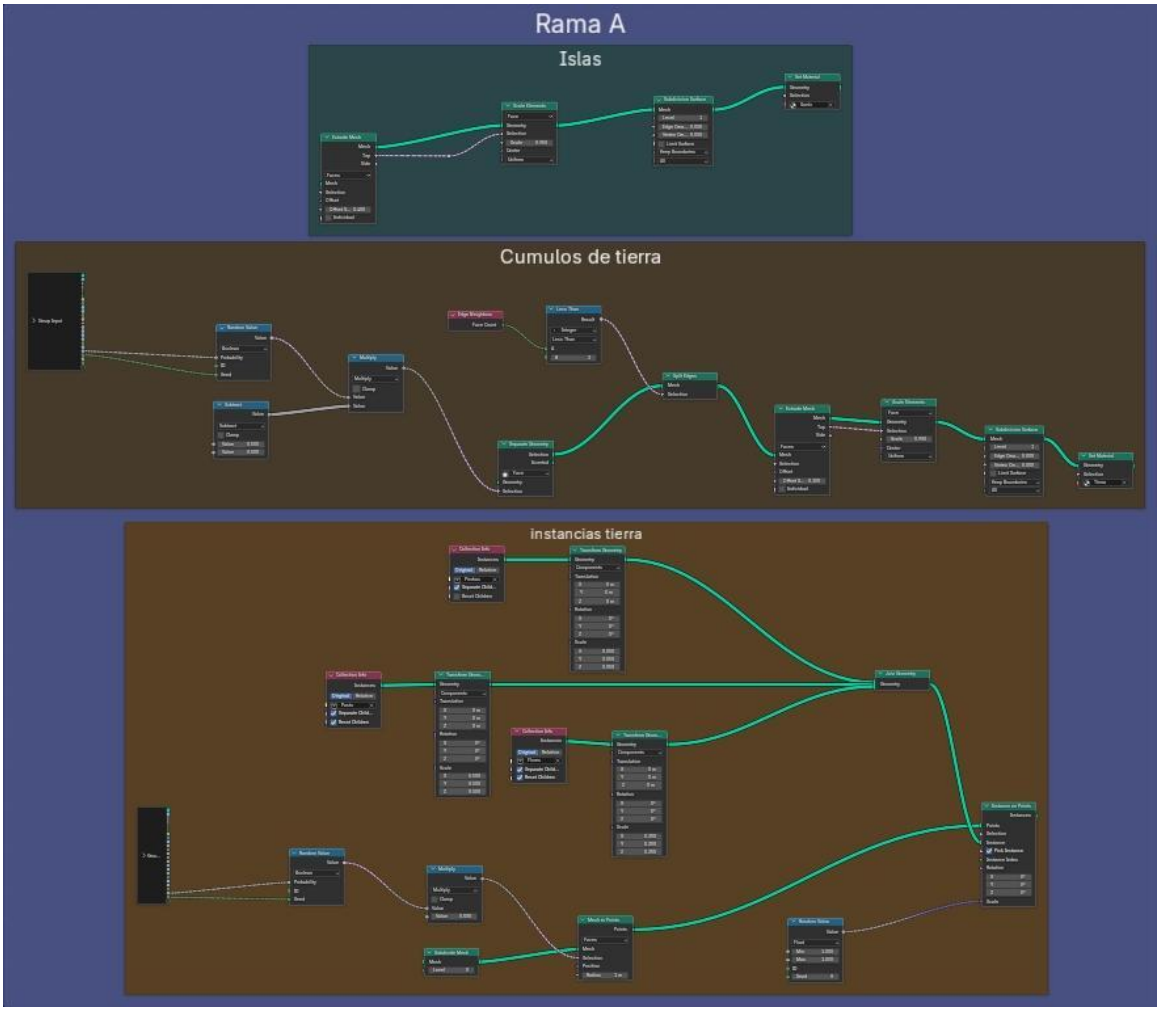


Figura 4. Rama A de nodos que generan el terreno.

Rama B: Generación de edificios

La rama B utiliza las caras que no fueron clasificadas como cúmulos de tierra para generar la arquitectura principal. Cada cara seleccionada se interpreta como la base de un edificio y a partir de estas huellas se aplica una secuencia de extrusiones y transformaciones geométricas que construyen el volumen de cada edificio. Estas extrusiones cumplen funciones distintas dentro del modelo: primero definen la base del edificio, después generan el cuerpo principal, posteriormente introducen variaciones de escala o ensanchamiento, y finalmente construyen remates. Este modelo permite que los edificios no sean simples prismas verticales. Durante esta etapa, el sistema conserva información sobre las caras laterales, superiores y zonas negativas de los edificios. Esta información se utiliza para derivar dos ramas adicionales: una para la colocación de ventanas y otra para la distribución de escaleras con portales. La rama de ventanas detecta las caras verticales de los edificios mediante la orientación de sus normales. Una vez identificadas las caras, se generan puntos sobre ellas y se instancian ventanas. Como la detección se basa en la geometría resultante, las ventanas se actualizan automáticamente cuando cambia la altura, el ensanchamiento o la forma del edificio. Esto evita colocar ventanas manualmente y garantiza que siempre permanezcan vinculadas a los edificios generados.

La segunda rama utiliza el espacio negativo de la ciudad, es decir, las zonas que no fueron ocupadas por edificios ni por cúmulos de tierra, para colocar escaleras con portales. Estas zonas funcionan como áreas de

circulación de la ciudad. En ellas se instancian diferentes escaleras con portales, los cuales también se generaron de forma procedimental. (Figura 5)

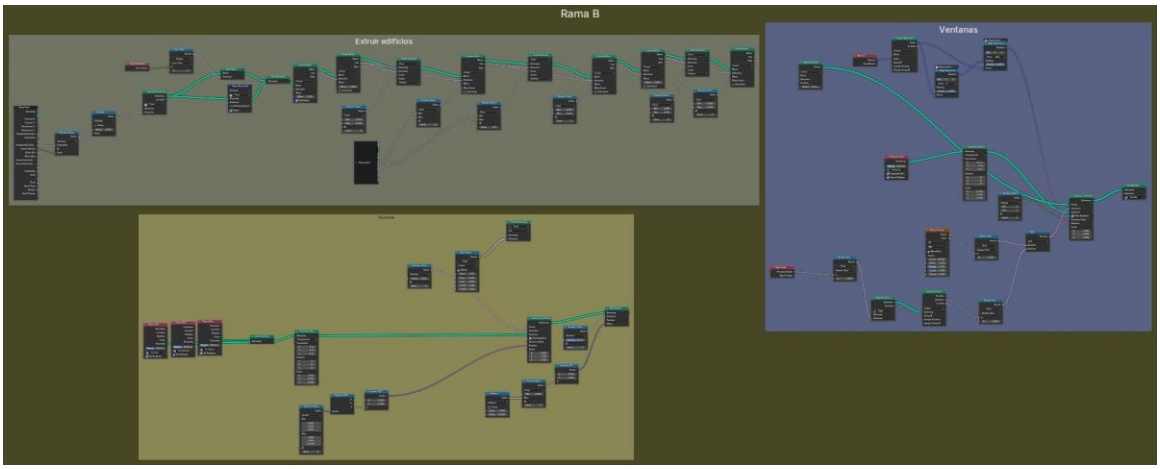


Figura 5. Rama B de nodos que generan los edificios, sus ventanas, y portales.

Control paramétrico y actualización en tiempo real

El sistema completo fue diseñado para funcionar de manera paramétrica. Aunque el árbol contiene muchas operaciones internas, solo se exponen al usuario los parámetros principales: semillas aleatorias, altura mínima y máxima de edificios, grado de ensanchamiento y valores relacionados con la densidad o variación de los islotes principales esto con la finalidad de no abrumar al usuario que vaya a construir su ciudad.

Al modificar cualquiera de estos parámetros, Blender recalcula el árbol de nodos y actualiza todos los elementos dependientes. Por ejemplo, si cambia la semilla, se redistribuyen los islotes, edificios y cúmulos; si cambia la altura de los edificios, las ventanas se reubican sobre las nuevas fachadas. La ciudad, por lo tanto, no es una colección de objetos colocados manualmente, sino un sistema de relaciones geométricas donde cada elemento deriva de una regla anterior.

Vegetación y rocas

Como complemento a la edificación, se generan también flores tipo amapolas, pasto y rocas de forma procedural. Para la creación de estos elementos se utilizaron Nodos de Geometría para su absoluta creación, no se hizo modelado poligonal estático, sino que, para los pétalos de las flores, el posicionamiento y tamaño del tallo, el pistilo fue hecho para que a partir de la PCG nazcan y no sean absolutamente iguales para todos. De igual forma, para la creación de pastos, absolutamente todo fue a partir de nodos procedurales para modelar y controlar la densidad de dichas instancias. Dentro del mundo se implementaron 5 conjuntos de cada uno de los elementos con diferentes características. Para la creación de las rocas y dentro de los nodos utilizamos 2 en particular, el primero de ruido (Noise) y el segundo de Voronoi, que fueron los que permitieron generarles esa textura tan particular a las instancias, como se puede observar en la Figura 6.

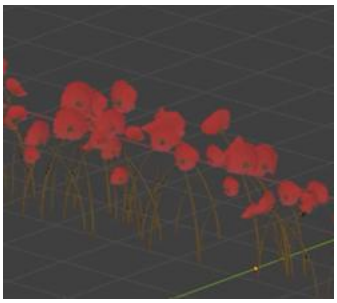


Figura 6. Visualización de las flores de amapola generadas de forma procedural.

Música y ambientación

Para lograr la generación y reproducción de música en tiempo real, la ambientación se diseñó a partir de tres softwares diferentes: CSound un lenguaje de programación de síntesis y procesamiento digital de sonido con la documentación de su página web (Csound, s.f.) encargado de la generación del algoritmo central procedural, loopMIDI el cable virtual que transmite información entre el algoritmo y el DAW (Digital Audio Workstation) REAPER como el sintetizador de audio.

Componer secuencias de notas basadas en las probabilidades de que una nota determinada esté precedida de otra u otras, técnica conocida como Cadenas de Markov. (Miró, 2017). Como primer paso hacia la creación en tiempo real de música procedural para la ambientación del proyecto se diseñaron matrices de transición donde cada fila representa el estado actual y las columnas representan la probabilidad de saltar a otra, o en este caso también se agregó la posibilidad de no tocar ninguna nota. Al ejecutar el código se definen la salida de audio, en este caso los Meta Quest 3S, y la salida hacia el puerto virtual de loopMIDI para la conexión con el DAW REAPER. Posteriormente, se establecen las configuraciones de audio como lo son: la frecuencia de muestreo (44100 Hz), el número de canales de salida, así como una semilla en base al reloj para generar una composición de la música diferente en cada ejecución.

Una vez definidas las variables globales de las configuraciones de sonido, se definen tres arreglos que contienen las notas musicales que se van a tocar. El primer arreglo contiene las notas base de la escala do menor y las otras dos corresponden a la tercera menor y a la quinta justa de cada nota para la creación de los acordes. A continuación, se definen las matrices de transición y las probabilidades de pasar a una nota o de no ejecutar ninguna nota. Para la elección de la siguiente nota se crea una variable de metrónomo que da la orden de ejecución cada 3 segundos, así no se empalman las notas con tiempo de duración prolongado como los son los acordes o los bajos, cuando el metrónomo acciona la ejecución en secuencia “gira” tres ruletas, para la elección de las notas que se van a enviar a REAPER, las tres ruletas funcionan de la misma manera, se genera un número aleatorio entre 0 y 1, este número se guarda en una variable que se va a comparar con las probabilidades de cada nota, si el número generado es menor que la probabilidad de la primera nota, se elige esa nota si no es así esa probabilidad se suma a la siguiente y se vuelve a comparar, así hasta elegir una nota, ésta nota se guarda y se envía a REAPER y se reinician los valores de todas la variables a cero para la siguiente nota.

Aunque menos complejos, los audios de viento, campana y ruido blanco también tienen su ejecución procedural. La ejecución se basa en un número pseudoaleatorio generado con una función que indica el tiempo de ejecución, así como cuánto tiempo hay entre una ejecución y otra. Además, para la campana se considera una variable más, esta de igual manera se basa en un numero pseudoaleatorio entre 1 y 100 generado por la función random, esta función se compara con un numero predefinido, en nuestro caso el número 15, si el numero genera es menor a 15 se ejecuta el sonido de campana, esto para condicionar a una baja probabilidad de un sonido y solo se genere de manera espontánea en cada ejecución.

Las decisiones tomadas por el algoritmo se envían a través de un puerto virtual, en este caso, LoopMIDI de Tobias Erichsen. Este funciona solo como un intermediario entre CSound y el DAW. REAPER recibe la señal enviada y la distribuye en distintos tracks de audio, donde se le agregan VST/VSTi (instrumentos virtuales), en este caso VITAL, para generar sintetizadores acordes a la inmersión de la experiencia. Además de agregar los instrumentos virtuales, se les asignan efectos sonoros como: ecualizadores, reverberaciones y demás

efectos de arreglos sonoros, para sintetizarlos y que se escuchen a la par con los elementos visuales de la ciudad procedural y demás instancias programadas.

Implementación

La realidad virtual es una tecnología diseñada para engañar al cerebro y hacerle creer que estás físicamente presente en un lugar generado por computadora, en lugar de la habitación donde realmente estás. Para traducir esa información del mundo generado por computadora, se necesita un traductor de información como lo es OpenXR.

OpenXR es un estándar de software creado para unificar la industria de la realidad virtual y la realidad aumentada. OpenXR actúa como una capa de abstracción que gestiona tanto la entrada de datos de telemetría desde el hardware como la salida visual renderizada desde el motor gráfico hacia los visores. OpenXR actúa como el intérprete simultáneo perfecto. Blender le entrega los gráficos y los cálculos a OpenXR, y OpenXR se encarga de traducirlos al lenguaje exacto que necesitan los lentes de Meta.

Para la visualización e inmersión de “Umbral de un Corazón Nostálgico”, se optó por la utilización de los visores de realidad virtual Meta Quest 3. La comunicación entre este hardware y el entorno de desarrollo de Blender se estableció a través de la aplicación oficial de Meta Quest Link, la cual actúa como puente para procesar el monitoreo básico de los controles y el posicionamiento del usuario. Sin embargo, el movimiento por defecto presentaba un desplazamiento libre que permitía alteraciones en el eje vertical, provocando que el usuario flote sobre la geometría. Para corregir este error, se desarrolló un script en Python directamente en Blender. La función de este código es restringir las coordenadas verticales, bloqueando el eje Z para anclar la cámara al nivel del suelo. Al operar en conjunto con la malla de colisión, este sistema garantiza una caminata realista y físicamente coherente por las ruinas de la ciudad.

Resultados

La implementación del sistema permitió obtener una ciudad tridimensional generada de manera procedural y controlada mediante parámetros modificables. El resultado final integra la construcción del terreno, la distribución de edificaciones, la colocación de vegetación, escaleras, portales y otros elementos ambientales dentro de un mismo flujo de trabajo desarrollado con Nodos de Geometría en Blender. El sistema no produce una única configuración fija, sino diferentes variaciones de la ciudad a partir de la modificación de parámetros como la semilla pseudoaleatoria, la densidad de los elementos y las dimensiones de las estructuras. Esto permitió conservar una identidad visual común entre los resultados, manteniendo al mismo tiempo diferencias en la distribución espacial y en la composición de cada ciudad generada.

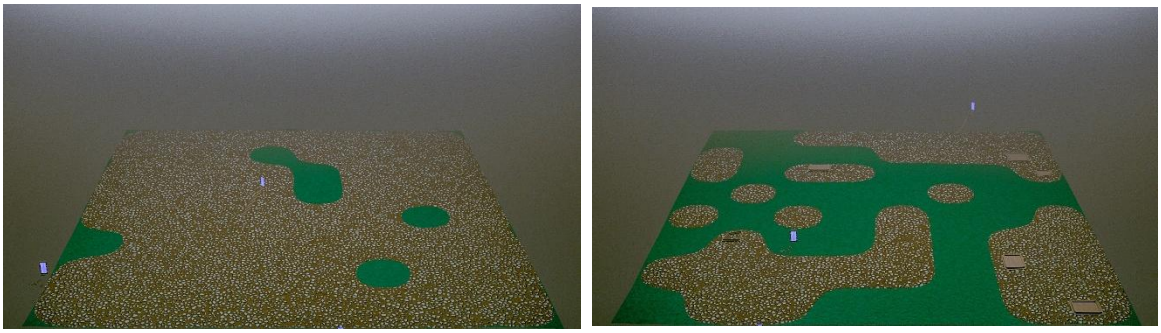
Distribución espacial del terreno

La ciudad obtenida está construida a partir de una geometría base que se modifica mediante operaciones matemáticas y selecciones pseudoaleatorias. Como resultado, se generaron regiones fragmentadas que funcionan como islotes independientes, sobre los cuales se distribuyeron las construcciones y los elementos naturales. La incorporación del diagrama de Voronoi y de funciones de ruido permitió introducir irregularidades en la geometría inicial, evitando que la ciudad conservara una estructura completamente uniforme. Estas modificaciones produjeron variaciones en la forma, el tamaño y la distribución de los islotes, reforzando la apariencia fragmentada y surrealista planteada en la propuesta visual. A partir de las regiones generadas, el sistema separó las zonas destinadas a edificaciones, terreno, vegetación y circulación. Esta organización permitió controlar la ubicación de los elementos sin necesidad de colocarlos manualmente uno por uno.

Uno de los principales resultados del sistema fue la creación de una interfaz paramétrica simplificada, denominada Ciudad, desde la cual el usuario puede modificar la estructura general de la ciudad sin intervenir directamente en el árbol de Nodos de Geometría. Los controles se organizaron en dos grupos: generación de islas y generación de edificios.

Para la generación de las islas se habilitaron los parámetros Probabilidad Islas y Seed Islas. El primero admite valores entre 0.5 y 0.9 y controla la proporción de superficies conservadas para conformar la estructura fragmentada del terreno. Los valores cercanos al límite inferior producen una mayor ocupación del espacio,

mientras que los valores altos conservan una menor cantidad de superficie. Seed Islas permite modificar la configuración espacial sin alterar las demás reglas del sistema (ver Figura 7).



Figuras 7. Distribución de islotes.

Distribución de instancias

El sistema permitió incorporar modelos previamente creados como instancias dentro de la ciudad. Entre estos elementos se encuentran edificios, vegetación, escaleras, portales y objetos complementarios del entorno. El uso de instancias permitió reutilizar una misma geometría en diferentes puntos, variando su posición, rotación y escala. Esto hizo posible aumentar la diversidad visual sin duplicar de manera independiente todos los objetos dentro de la escena. Las reglas espaciales restringieron la aparición de cada tipo de elemento a regiones determinadas. La vegetación fue distribuida sobre las zonas de terreno seleccionadas, mientras que las construcciones se colocaron en las superficies destinadas a los “lotes urbanos”. Las escaleras y los portales fueron integrados en zonas transitables.

En la generación de edificios se habilitaron controles para modificar su probabilidad de aparición, semilla, altura mínima, altura máxima y ensanchamiento. Probabilidad Edificios, con un rango de 0 a 0.15, determina la cantidad de construcciones distribuidas sobre las regiones disponibles. Seed Edificios produce diferentes distribuciones manteniendo constantes los demás parámetros. La altura de las construcciones se controla mediante Altura Min, con valores entre 3 y 10 unidades, y Altura Max, cuyo valor inicia en 3 y puede incrementarse hasta el límite numérico admitido por Blender. Ensanchamiento Min admite valores de 0.1 a 1, mientras que Ensanchamiento Max puede variar entre 0.1 y 5. La combinación de estos controles permite generar ciudades con distintos niveles de densidad, verticalidad y variación volumétrica. Para conservar configuraciones coherentes, el valor de Altura Max debe ser igual o mayor que Altura Min, y Ensanchamiento Max debe ser igual o mayor que Ensanchamiento Min. Las semillas no se describen como infinitas, sino como valores enteros dentro del rango numérico permitido por el sistema. La modificación de estos controles produjo configuraciones diferentes sin cambiar la estructura interna del generador. Las semillas alteraron la distribución espacial, mientras que las probabilidades y los rangos dimensionales modificaron la ocupación, la altura y la amplitud de las construcciones. De esta manera, el usuario puede explorar distintas composiciones manteniendo una identidad visual común (ver Figura 8).

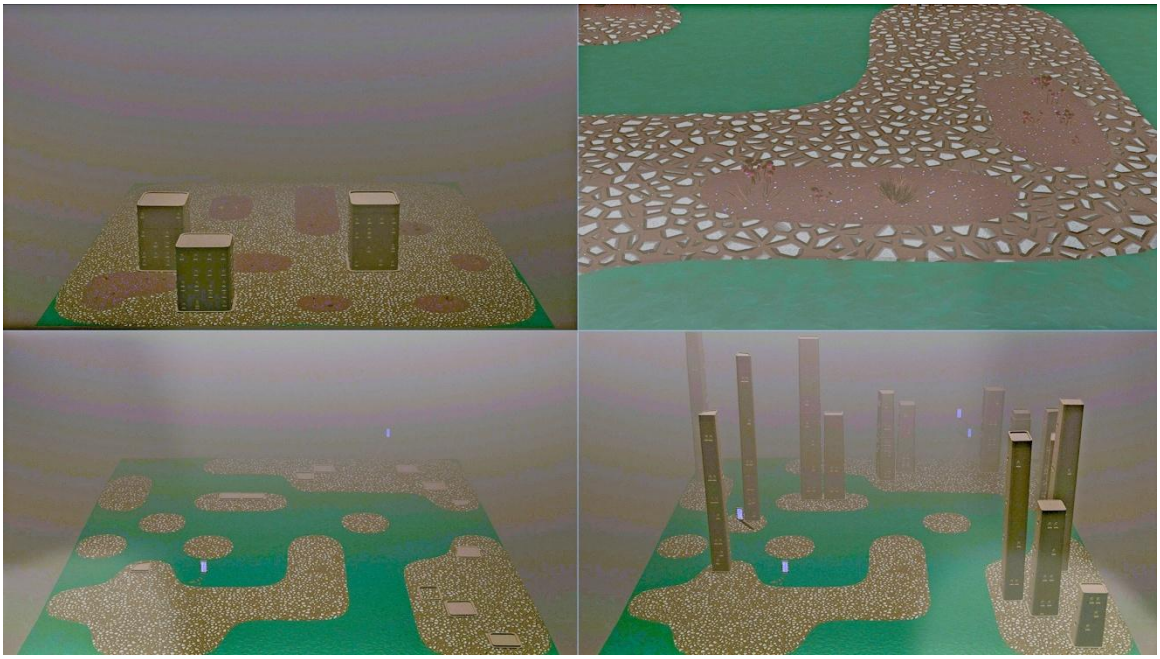


Figura 8. Comparación entre la geometría base, la división espacial y el resultado general de la ciudad.

Integración musical

La síntesis de música en tiempo real se logró a través de un código en CSound (Figura 9) el cual funciona como el cerebro, mandando las notas a través de un cable MIDI hacia un DAW que ejecuta las notas componiendo la música con notas establecidas en la escala de do menor, así como también sus correspondientes en tercera menores y quintas justas para realizar acordes para instrumentos como sintetizadores. Cada una de estas notas se representa con un numero en el estándar MIDI como se muestra en la tabla 1.

***Tabla 1**. Numeración de las notas en el estándar MIDI.*

Numero MIDI	Nota	Nombre en español	Tercera Menor	Quinta Justa
48	C3	Do	51	55
50	D3	Re	53	56
51	D#3	Re sostenido	55	58
53	F3	Fa	56	60
55	G3	Sol	58	62

56	G#3	Sol sostenido	60	63
58	A#3	La sostenido	62	65
0	—	Silencio/ nota apagada	0	0

El resultado del uso de la escala en do menor crea una melodía triste o nostálgica, encajando en perfecta armonía con la atmósfera de nuestro entorno tridimensional. De esta manera se aumenta la sensación de inmersión de tensión e incertidumbre, respaldada por el diseño sonoro de viento, ruido blanco y la campana que acompañan durante el tiempo de experiencia limitado a 5 minutos.

```
giMarkovMatrix[] fillarray \
  0.10, 0.15, 0.10, 0.15, 0.20, 0.10, 0.05, 0.15, \
  0.10, 0.10, 0.15, 0.20, 0.20, 0.10, 0.05, 0.10, \
  0.15, 0.10, 0.10, 0.20, 0.10, 0.15, 0.05, 0.15, \
  0.25, 0.10, 0.10, 0.10, 0.25, 0.05, 0.05, 0.10, \
  0.30, 0.05, 0.05, 0.10, 0.15, 0.15, 0.05, 0.15, \
  0.15, 0.20, 0.10, 0.15, 0.15, 0.10, 0.05, 0.10, \
  0.40, 0.10, 0.10, 0.10, 0.10, 0.10, 0.00, 0.10, \
  0.25, 0.15, 0.10, 0.15, 0.15, 0.10, 0.10, 0.00


giOrganMarkovMatrix[] fillarray \
  0.50, 0.05, 0.10, 0.05, 0.20, 0.05, 0.05, 0.00, \
  0.40, 0.10, 0.10, 0.10, 0.20, 0.05, 0.05, 0.00, \
  0.30, 0.05, 0.20, 0.10, 0.30, 0.05, 0.00, 0.00, \
  0.40, 0.10, 0.10, 0.10, 0.20, 0.05, 0.05, 0.00, \
  0.30, 0.05, 0.10, 0.05, 0.40, 0.05, 0.05, 0.00, \
  0.40, 0.05, 0.10, 0.10, 0.25, 0.05, 0.05, 0.00, \
  0.50, 0.05, 0.10, 0.05, 0.20, 0.05, 0.05, 0.00, \
  0.40, 0.10, 0.10, 0.10, 0.20, 0.10, 0.00, 0.00


giVocalsMarkovMatrix[] fillarray \
  0.10, 0.05, 0.05, 0.10, 0.10, 0.05, 0.05, 0.50, \
  0.05, 0.10, 0.05, 0.10, 0.10, 0.05, 0.05, 0.50, \
  0.10, 0.05, 0.10, 0.05, 0.10, 0.05, 0.05, 0.50, \
  0.05, 0.10, 0.05, 0.10, 0.10, 0.05, 0.05, 0.50, \
  0.10, 0.05, 0.10, 0.05, 0.10, 0.05, 0.05, 0.50, \
  0.05, 0.10, 0.05, 0.10, 0.10, 0.10, 0.00, 0.50, \
  0.10, 0.05, 0.10, 0.05, 0.10, 0.05, 0.05, 0.50, \
  0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.10, 0.30
```

Figura 9. Matiz de transición de las cadenas de Markov en Csound.

Resultado general de la experiencia

La integración de los diferentes componentes dio como resultado una experiencia audiovisual inmersiva formada por una ciudad paramétrica, elementos instanciados, una propuesta narrativa y visual, una composición musical generativa y su visualización mediante realidad virtual. El sistema desarrollado demuestra que una misma estructura procedural puede producir diferentes configuraciones espaciales conservando reglas comunes de composición. Además, la posibilidad de modificar los parámetros en tiempo real facilitó la exploración de distintas alternativas visuales sin rehacer manualmente la ciudad.

Conclusiones

En este documento se presentó el desarrollo de una experiencia inmersiva de Realidad virtual realizada a través de PGC. Esta experiencia explora la creación de un entorno tridimensional con Nodos de Geometría integrados en Blender utilizando algoritmos como los diagramas de Voronoi. Además, acompañado de música sintetizada en tiempo real en CSound basada en las cadenas de Markov. Además, se integraron e implementaron en VR con los Meta Quest S3. El título de la experiencia es “Umbral de un Corazón Nostálgico”. Con esta exploración verificamos que la PCG se vuelve una alternativa viable para la creación de entornos de gran densidad, como en nuestro caso ciudades enteras con vegetación, cuerpos de agua y estructuras fantásticas como lo son los portales, aún más cuando se busca que la experiencia sea única para

cada usuario. La creación de entornos tridimensionales se puede utilizar en videojuegos o películas que lo requieran. Aunque claro esta contiene limitaciones en especial de rendimiento, puesto que al realizar cálculos computacionales y actualizarlos constantemente genera una gran carga de información para las computadoras, por lo que no es fácil acceso. Aunque con tiempo limitado, se logró una gran experiencia inmersiva. Como trabajo a futuro, es posible explorar más variables para crea un ambiente sea único para cada usuario, entre las ideas está el usar la cámara de los Meta Quest S3 para modificar la música en base a lo que veía: mientras más densidad de elementos existían en su campo visual la música contenía más elementos, por lo contrario, si la densidad era menor los instrumentos desaparecían. Además, es necesario crear una versión con mejor rendimiento computacional o crear una alternativa con un video 360 grados para una demostración del proyecto de mayor acceso.

Referencias

Aurenhammer, F. (1991). Voronoi diagrams- A survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR), 23*(3), 345 - 405. doi: https://doi.org/10.1145/116873.116880

Csound. (n.d.). *Csound*. Retrieved from Free and Open Source Audio Programming Language: https://csound.com/get-started.html

Hamdani, A. (2026). *Hamdani, Abdelilah. 3D Environment Design with Blender 5: Enhance your modeling, texturing, and lighting skills to create realistic 3D scenes.* Packt Publishing Ltd.

Hello Games. (2016). *No Man's Sky*. Retrieved from [Videojuego].

Lens, S. (2023). *Procedural 3D modeling using geometry nodes in Blender: discover the professional usage of geometry nodes and develop a creative approach to a node-based workflow.* Packt Publishing Ltd.

Miró, S. R. (2017). *Composición musical asistida por ordenador.* Retrieved from [tesis de licenciatura, Universidad de Alicante]: https://grfia.dlsi.ua.es/repositori/grfia/degreeProjects/31/ribes-report.pdf

Orrego, C. S. (2019). *"Diseño de Algoritmo de Generación Procedural".* [Tesis de licenciatura, Universidad Técnica Federico Santa María].

Sánchez, C., & Romero, A. (2018/2019). *Creación de mundos mediante generacion procedural de Unity*. Retrieved from [Tesis de licenciatura, Universidad Complutense de Madrid].

Studios, Mojang. (2011). *Minecraft*. Retrieved from [Videojuego]: minecraft.net

van Gumster, J. (2022). Procedural Modeling with Blender's Geometry Nodes: A workshop on taking advantage of the Geometry Nodes feature in Blender for procedural modeling. *ACM SIGGRAPH 2022*, (pp. 1-2).