Universidad de las Ciencias Informáticas

Universidad de las Ciencias Inform?ticas
Centro de Tecnologías Interactivas, Facultad 4

Elementos necesarios para desarrollar un editor procedural de estructuras de ciudades usando Unity

Informe técnico para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Adrian Preval González

Tutores: Msc. Luis Guillermo Silva Rojas

Ing. Laura Elena Santana Rojas

La Habana, 2020

Agradecimientos

Quiero aprovechar esta sección del documento para agradecerle a todas las personas que han formado parte de este logro y que han brindado su apoyo en el transcurso de estos años.

A los primeros que deseo agradecer son los profesores que me han guiado a lo largo de estos 5 años, y me han nutrido de las herramientas necesarias para afrontar mi futuro profesional. Especialmente quisiera agradecerles a mis tutores Laura Elena y Guillermo que a pesar de lo atípico de esta situación siempre han estado ahí para ayudarme con todo lo relacionado a la tesis, además de estar pendientes a su desarrollo. También quisiera agradecer a la paciencia que me ha tenido la profesora Daylen en todo el proceso de revisado de tesis, y se que he sido un dolor de cabeza para ella.

A mis amigos del Papiparty, que nos conocemos desde la secundaria y hemos estado siempre juntos a lo largo de los años, de aquí quisiera mencionar en especial a Conrrado, El Papi, Edney, Mimi,YY y Raymel. También a las amistades que he conocido en la UCI, a mi grupo y todas sus re-estructuraciones, al Consejo y la extrañación, quienes se han colado en mi corazón, especialmente a Naisel, Asley, Chris, Nai, Loy, Lia, Rai y mi arrastre Cristhian. Por último, quiero mencionar a un grupo de personas que he conocido en el transcurso de la cuarentena a través de grupos de Whatsapp y que me han acompañado y entretenido en estas largas noches, en especial Ari, Pedro, Mely, Rhae, Dash, Poe, Xaly y Vivi.

También quisiera mencionar a la parte más importante de mi vida y que me han formado como persona, mi familia, la que me ha enseñado los pasos a seguir en mi recorrido por el mundo, y el pie de apoyo fundamental en el transcurso de mi carrera. Quisiera señalar que no toda la familia posee vínculo sanguíneo, esto me lo han demostrado una familia de vecinos que me han brindado todo su amor desde chiquito, con la cual he formado una relación inquebrantable. Agradecer también a mi Papa que siempre ha aportado su granito de arena en el desarrollo de mi ser, mi hermanita pequeña que es la artista de la familia y ser de luz, a mis primos queridos,

a Luisi creo que existen pocas personas que brinden su amor y apoyo de una manera tan desinteresada como él, a mí abuelo Illo que aunque ya no se encuentre en este mundo siempre agradeceré todo lo que hizo por mí, a mis abuelas, Mercedes que nos adoramos aunque hallan muchos Km entre nosotros, e Isabelita la persona que más me consiente sobre la faz de la tierra haciéndome sentir siempre como una persona especial y el número uno para ella.

De último quisiera guardar la posición para la persona que más me ha apoyado a lo largo de mi vida, la principal responsable de que yo ingresara en la universidad y terminara esta carrera, un eslabón invaluable en mi desarrollo personal, mi madre, Lisette Isabel González Martínez, la persona más increíble que conozco.

Resumen

Actualmente, la reconstrucción de ciudades en 3D se utiliza ampliamente en temas importantes como el diseño urbano, la simulación de tráfico y la creación de videojuegos. La generación procedimental de ciudades ha sido tema de un gran número de investigaciones en los últimos años, como una forma para el ahorro de tiempo en la producción de diseños en 3D. En este trabajo se realiza una caracterización de los principales conceptos asociados a la generación procedural de estructuras de ciudades. Se identificaron las principales investigaciones existentes sobre la generación de ciudades de forma procedural, se brinda una descripción general de sus implementaciones y una crítica de su funcionalidad y resultados, lo que permitió plasmar las fortalezas y limitantes de cada uno de los algoritmos. Se recomida el uso del método L-Systems para la generación de mallas poligonales, por las ventajas descritas en la investigación. Como estructura de datos subyacente, se recomiendan los grafos conexos, por los datos que aporta en la conformación de la red. Adicionalmente, se propone el diseño inicial de un módulo para Unity que facilite la edición dinámica de estructuras básicas de ciudades para videojuegos.

Palabras clave: 3D, calles, ciudades, gráficos, mallas, procedural, técnicas, triangulación, Unity, videojuegos.

Índice general

Índice de figuras

Introducción

La pintura es una de las expresiones artísticas humanas más antiguas y una de las llamadas Bellas Artes, considerada una de las expresión más identificativa del concepto humano. Las primeras representaciones de imágenes se encuentran en las pinturas rupestres, donde el hombre reflejaba su existencia y su interacción con el entorno. La sensación de realismo lo obtenían por medio del volumen, aprovechando el abultamiento natural de la roca, con movimiento y policromía, se cree que estas pinturas cumplían un ritual mágico [1]. Siglos después, el hombre, agrupado en sociedad, desarrolla el arte pictórico con representaciones de su forma de convivencia y sus creencias. Los egipcios pintaban las tumbas de sus faraones con representaciones mitológicas y escenas de la vida cotidiana, como la caza, la pesca y las celebraciones [2]. Para el XV en el norte de Italia se introdujo el uso del lienzo, que es una especie de tela con un entramado característico del hilo. Con el pasar de los años el lienzo se ha convertido en el soporte preferido de muchos artistas de la plástica, sobre el cual emplean distintas técnicas, entre las más usadas se encuentra el óleo [1].

Como parte de la evolución del hombre, el aumento de las tecnologías y hallazgos científicos, surgieron dos de los avances más importantes logrados por el ser humano moderno, la fotografía y la impresión. La fotografía permitiría la captura de imágenes de forma duradera y la impresión brinda la posibilidad de reproducir manuscritos y libros para la posteridad [3]. En 1957, Russell Kirsch, un científico del National Institute of Standards and Technology creó la que se considera la primera imagen digital, recreada (no producida) en píxels. Para ello, Kirsch utilizó un proto-escáner que transformó la imagen de su hijo de tres meses en una matriz de ceros y unos, permitiendo visualizarla en la Computadora Automatica Electronica de Standars (SEAC por sus siglas en inglés) [4].

La imagen digital es la representación bidimensional de una imagen empleando bits, unidad mínima de información compuesta por dígitos binarios (1 y 0), que son los que regulan los sistemas informáticos y cualquier dispositivo de tipo digital. El potencial de los conceptos de la imagen digital no se dió hasta que se combinaron los primeros grandes ordenadores digitales con los intereses de la exploración espacial y los avances médicos en la década de 1960. Fue entonces cuando empezaron a probarse técnicas de manipulación de imágenes electrónicas, utilizando el ordenador para mejorar las imágenes recibidas por las sondas espaciales [5].

En los años 70 se produce un importante salto cualitativo y cuantitativo en la tecnología de la ima-

gen digital con el desarrollo de los microprocesadores, que permitieron no sólo ampliar la capacidad de procesamiento y almacenamiento de imágenes, sino también el desarrollo de los dispositivos de carga acoplada (CCD, por sus siglas en ingles), un sensor con diminutas células fotoeléctricas capaces de registrar la imagen. Efectivamente el CCD fue fundamental para el desarrollo de las cámaras digitales y es aún un componente fundamental en ellas, en cualquiera de sus formas [6]. Debido a la progresión de los procesadores y su influencia en la mejora los gráficos por computadoras se realizó la inclusión de estos en la televisión, gracias a los sistemas de dotación física creados por Computer Image Corporation (CIC). En 1977 la industria de los videojuegos experimentó de nuevo una revolución, esta vez fue el acelerador 3DFX Voodoo 3D. Este chip 3D aplastó por completo a la competencia con su increíble y extremadamente práctico desempeño en 3D, dando así un salto evolutivo en los gráficos por computadora [7].

Con la evolución de los gráficos asistidos por computadora, el término diseño 3D pasó a ser casi de uso exclusivo del lenguaje de los computadores, Autocad, ha sido el programa ícono de diseño 3D, pues en sus previas versiones ya proyectaba los planos técnicos y abría poco a poco la posibilidad de un entorno 3D, permitiendo visualizar los proyectos ideados desde el papel en entornos 3D. Este hecho cambió las normas del juego, ocasionando que empezara una nueva disciplina y campo profesional, el de los modeladores y diseñadores 3D [8]. Hoy en día son innumerables las disciplinas y profesiones que requieren de la técnica del diseño 3D, entre ellas: la arquitectura, la ingeniería, el diseño industrial, la animación 3D, la geometría descriptiva, el diseño gráfico, entre otras, incluso las artes plásticas se aproximan a lo tridimensional de varias maneras.

En los últimos años debido al aumento en las exigencias sobre la calidad de los productos de la industria del entretenimiento, por parte de los consumidores, salió a flote una nueva tecnología para la generación de ambientes virtuales de una forma más ágil, la Generacion Procedural de Contenido (PCG, por sus siglas en ingles). Con PCG se puede aliviar la carga de producción, con solo cambiar los parámetros del generador se obtiene una gran cantidad de contenidos similares. En ausencia de técnicas de PCG el modelado manual es un trabajo exhaustivo que debe ser hecho por un diseñador [9].

A lo largo de los años, los investigadores han realizado considerables progresos hacia el desarrollo de técnicas eficientes, contemplando paisajes naturales cubiertos de vegetación [10] y ciudades [11]. Los algoritmos procedurales se han desarrollado para generar grandes ciudades con complejas redes de calles [12]. La creación de modelos convincentes es una tarea crucial en la industria del entretenimiento y diversas aplicaciones de formación y planificación urbana. Sin embargo, el modelado y visualización de sistemas artificiales como grandes ciudades es un gran desafío para los gráficos por computadora, modelar los detalles de entornos urbanos tridimensionales consume mucho tiempo y puede requerir varios años de trabajo. Una solución poderosa para la modelación urbana a gran escala es el uso de tecnología procedimental [13].

El desarrollo de las tarjetas gráficas, los algoritmos para generar gráficos por computadoras y el desarrollo de potentes herramientas para el modelado 3D, han permitido representar escenas tridimensionales con alto nivel de precisión y detalle, pudiendo generar entornos completos con mucha semejanza a la realidad. Las escenas generadas se emplean usualmente en aplicaciones de realidad virtual, realidad aumentada, en la industria o en el área de los videojuegos. A pesar que el desarrollo del videojuego y el uso de los gráficos por computadoras en Cuba empezó desde los años noventa de la mano de instituciones como la Universidad de las Villas y los Joven Club de Computación [14], no ha sido hasta estos últimos años que han tomado auge y popularidad en la isla, encabezado por el ICAIC y la Universidad de Ciencias Informáticas con el centro de tecnologías interactivas, VERTEX.

El centro VERTEX tiene dentro de sus líneas de investigación el Desarrollo de Paseos Virtuales, Realidad Aumentada y Videojuegos. A través de una entrevista con el cliente se puede definir que entre las principales dificultades que se encuentran en el centro están que al ser una industria básicamente incipiente en el país, este no cuenta con los recursos materiales para el desarrollo adecuado y no existe mucho personal especializado para la realización de diseño, guión y sonidos que conllevan la creación de videojuegos. Además las principales herramientas utilizadas en el desarrollo de estos son de carácter privativo y el pago de las licencias para el uso de estas es demasiado elevado teniendo en cuenta la situación económica del país, por lo que los escenarios se confeccionan de forma manual. Este proceso, en dependencia de la complejidad de las escenas, demora un tiempo considerable y no siempre se cuenta con el personal capacitado para crearlas. Específicamente en el área de los videojuegos, es común crear escenarios basados en ciudades, reales o ficticias, con su correspondiente estructura de calles, aceras, manzanas y edificaciones. El modelado manual de estas escenas emplea, en el centro VERTEX, un tiempo considerable, un ejemplo de esto se encuentra en el desarrollo del videojuego Kuba Kart, que tomaba alrededor de una semana el modelado de cada pista o nivel.

Teniendo en cuenta la situación problemática antes descrita, se identifica como problema de investigación ¿Cómo agilizar el proceso de creación de estructuras de ciudades para videojuegos orientados a dispositivos móviles? Se define como objeto de estudio: Proceso de generación de escenas 3D.

Para dar solución al problema de investigación planteado, se define como objetivo general: Caracterizar los elementos necesarios para el correcto desarrollo de un componente de Unity, que facilite la edición dinámica de estructuras de ciudades. Teniendo en cuenta el objetivo planteado, se define como campo de acción: Generación de mallas 3D de estructuras de ciudades en Unity.

A partir de este objetivo general se desglosan los siguientes objetivos específicos:

- Realizar un estudio acerca de los conceptos fundamentales de la generación procedural de carreteras.
- Realizar un estudio de las tecnologías, lenguajes y herramientas establecidas el desarrollo del modulo.
- Diseñar la propuesta del modulo.

Métodos teóricos

- **Histórico-Lógico:** permitió un estudio sobre las distintas herramientas que se han utilizado para el diseño de escenarios 3D así como de distintos métodos y algoritmos útiles para ello.
- **Analítico-Sintético:** se empleó para el análisis de teorías y documentos, extrayéndose cada uno de los aspectos esenciales de las herramientas y la literatura seleccionada para una correcta elaboración del componente.

Métodos empíricos

- **Entrevista:** permitió establecer un intercambio con el cliente para comprender sus necesidades, con el objetivo de de?nir el problema existente. A partir de la situación problemática y de capturar los distintos requerimientos que se necesiten para llevar a cabo la solución se podrán emitir las distintas historias de usuario.

CAPÍTULO 1

Desarrollo

En este documento se describen los principales conceptos asociados al objeto de estudio y las tareas de investigación. Se plantean las principales características que poseen los editores y generadores de ciudades. Además se especifican las herramientas y tecnologías que se utilizarán a lo largo de la investigación, así como la metodología que guiará este proceso.

1.1. Generación Procedural de Contenido

La Generación Procedural de Contenido (PCG, por sus siglas en ingles) es un término ampliamente utilizado en el desarrollo de videojuegos y el estudio de los gráficos por computadora. Según Mark Hendrick las PCG se identifican como el proceso de generar contenido virtual de forma automática o semi-automática [9], también Julian Togelius define a la PCG como algoritmos de generación de contenido virtual, con limitación humana [15]. Dado estos dos conceptos se puede definir la PCG como la generación de contenido virtual automatizada o semi-automatizada apoyada por la contribución humana. El término contenido virtual se refiere principalmente a objetos en una escena virtual, pero abarca también otros elementos como textura, animación, música, historia y mecánica de juego [9].

Se ha demostrado que el contenido generado de forma procedural en gráficos por computadora es beneficioso por varias razones. Ya sea para crear texturas, geometrías o efectos, los algoritmos de procedurales brindan variedad y complejidad en los detalles. Dado un pequeño conjunto de parámetros, un sistema de procedimiento puede generar un conjunto diverso de salidas complejas. Esto, según Ebert y col. se conoce como amplificación de datos [16]. Las implementaciones precisas pueden crear toneladas de contenido sin la necesidad de poseer grandes datos por adelantado. Como resultado, se pueden generar entornos ricos en detalles a partir de pequeños segmentos de código. Los resultados de contenido generado procedimentalmente ofrecen a los artistas libertad experimental. Ajustar cada parámetro puede resultar en posibilidades diversas. Por lo tanto, la generación procedural es un método poderoso en muchas áreas en las que se utiliza.

Según Ebert y col., hay tres propiedades claves para una generación procedural exitosa:

- **Abstracción:** los parámetros, los datos y el comportamiento se abstraen en algoritmos o procedimientos. El usuario debe necesitar una cantidad mínima de información sobre los detalles geométricos y la implementación.
- **Control paramétrico:** los parámetros deben responder a cada comportamiento específico para la generación. Podría haber varios parámetros para un control preciso de la salida, lo que permitiría al usuario crear libremente. Algunos ejemplos de parámetros podrían ser el detalle de la textura, la tasa de emisión de partículas, la altura del terreno, la profundidad del agua, el número de pisos de los edificios, etc.
- **Flexibilidad:** los parámetros no deben restringir los comportamientos en términos de restricciones del mundo real. Podrían generar valores poco realistas y producir resultados fantásticos. También se puede usar su imaginación para encontrar métodos de resultados atractivos irrelevantes a partir de un algoritmo de generación de procedimientos desarrollado para un propósito diferente.

Aunque el uso de las PCG ya está bastante establecido en el desarrollo de videojuegos, haciendo presencia en la creación de varios, aún tiene un amplio campo de investigación ya que las exigencias son cada vez mayores. Desarrollar un algoritmo de PCG que sea definitivo es bastante difícil, porque los recursos computacionales que se utilizan en los ambientes virtuales son cada vez más altos, también existen mayores expectativas por parte del consumidor exigiendo una elevada calidad gráfica. A estos problemas se les debe agregar el aumento del costo promedio de los nuevos proyectos y la creciente complejidad de las plataformas de desarrollo, haciendo así, que la progresión de estos algoritmos sea de vital importancia [17]. Según Julian Togelius, los principales objetivos a los que deben ser dirigidas estas investigaciones y mejoras serían: Multi-contenido, Diseño de juego y Generación de juegos completos [15]. Estos tres objetivos abarcarían de manera general la creación de un juego, sin embargo Hendrikx y col. hicieron un análisis de los distintos elementos que pueden ser generados en un juego de forma procedimental y extendieron estas tres características a 6 tipos diferentes, mostrados en la Fig 1.1 [18].

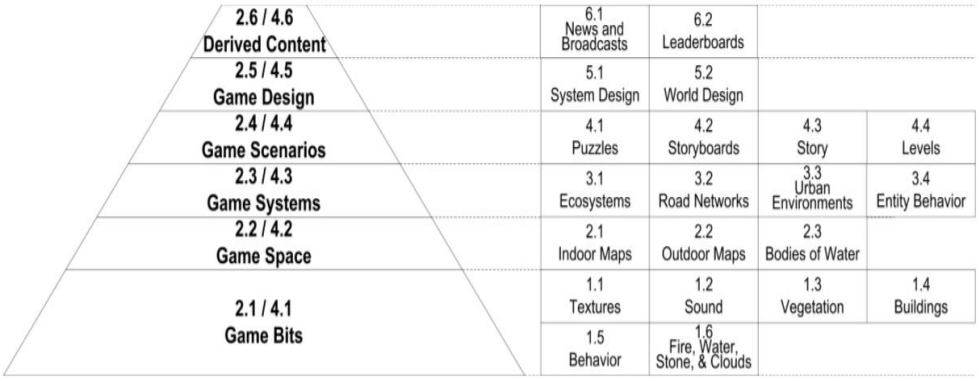

Figura 1.1. Escala de contenidos. Fuente: [18]

Estos 6 tipos de contenidos a generar engloban la creación de un juego en sentido general, partiendo desde las partes más generales del videojuegos que estarían en el escalón más bajo, hasta las partes más

especificas.

- **Game Bits:** son unidades elementales del contenido del juego, que normalmente no involucran al usuario cuando se consideran de forma independiente. Estas se diferencian entre sí como las partes del juego concretas y las abstractas: las unidades concretas como los árboles pueden ser elementos con los que interactuar en el mundo simulado, mientras que las unidades abstractas, como las texturas y el sonido, deben combinarse para producir una parte concreta.
- **Game Space:** Es el entorno en el que se desarrolla el juego y está parcialmente lleno de partes del juego entre las que navegan los jugadores. El espacio de juego puede definirse tanto de forma concreta como abstracta [19]. Los espacios concretos están estrechamente relacionados con la forma en que los humanos perciben; pueden ser bosques, laberintos, llanuras, etc. El tablero del ajedrez es un ejemplo de espacios de juego abstracto. El espacio del juego juega un papel importante en la creación de un juego interesante, ya que los jugadores a menudo construyen su interpretación del juego a partir del espacio del juego. [19]
- **Game Systems:** El uso de modelos y teoría de sistemas complejos para generar o simular partes de un juego no es infrecuente. Los sistemas de juego pueden ser abstractos, por ejemplo, sistemas que definen la relación entre la vegetación y las características de los mapas de exteriores, o concretos, como simulaciones de ciudades y redes de ciudades. El uso de sistemas de juego puede hacer que los juegos sean más creíbles y por lo tanto, atractivos. Muchos sistemas utilizados en los juegos son similares a los sistemas y modelos complejos de los libros de texto [20].
- **Game Scenarios:** Los escenarios del juego describen la forma y el orden en que se desarrollan los eventos del juego. Se pueden distinguir dos tipos de escenarios de juego, abstracto y concreto. Los escenarios abstractos del juego describen cómo se interrelacionan otros objetos. Los escenarios concretos del juego se presentan explícitamente en el juego, por ejemplo, como parte de la narrativa del juego [20].
- **Game Design:** El diseño de un juego se compone de contenido como reglas y metas; un componente estético, como un arco dramático o un tema gráfico, también son elementos importantes en el diseño [21]. Un juego puede verse como una instancia de un diseño de juego, en el que se han establecido los parámetros de las reglas y del objetivo. Un diseño de juego puede referirse a contenido de juego de todos los tipos descritos en la Fig.1.1, incluida la referencia recursiva a otro contenido de diseño de juego. El diseño de juegos se puede complementar con la generación (semi) automática de juegos o proporcionando herramientas que ayuden al diseñador a convertir ideas en contenido de diseño de juegos.
- **Derive Content:** Se define el contenido derivado como contenido que se crea como un producto secundario del mundo del juego. Hacer esto puede aumentar en gran medida la sensación de inmersión del jugador en el mundo del juego, ya que los jugadores registran sus experiencias dentro del juego para revisarlas dentro o fuera del juego [20].

El análisis que realizó Hendrikx y col. nos permite conocer las disimiles posibilidades que brinda la PCG

en la creación de contenido, además crea una escala donde se evidencia qué tan profunda es la generación de este. Para la creación del módulo solo se debe llegar hasta el nivel 3, que es donde se encuentra todo lo relacionado a los sistemas de juego, particularmente el sistema de juego concreto que engloba a la simulación de ciudades o de redes de carreteras. Pero no se debe olvidar los niveles 1 y 2 donde se define la creación del espacio de juego y las texturas.

1.1.1. Generación Procedural de Ciudades

Recientemente los investigadores de PCG han dirigido su atención a su aplicación en el contexto de la generación de fenómenos provocados por el hombre, como un área urbana. La generación de la ciudad se logra a través de una serie de etapas en las que cada una utiliza una serie de técnicas para crear carreteras, lotes, estructuras de edificios y caras de edificios.

Las ciudades son visual y funcionalmente complejas. Para obtener una mejor comprensión de estos sistemas complejos y sus estructuras, se puede observar en la investigación que identifica los patrones y los elementos constitutivos [22]. Los autores de diseño urbano y arquitectura han discutido una gran cantidad de temas de amplio alcance relacionados con los paisajes urbanos. De la investigación estudiada, se seleccionaron trabajos que se relacionan estrechamente con la estructura de la ciudad y presentan la correlación más directa con la generación de la ciudad. Dos publicaciones en particular cumplen estos criterios: Lynch escribe sobre la imagen de la ciudad y la percepción humana, detalla los elementos constitutivos de las ciudades como caminos, bordes, distritos, nodos y puntos de referencia [23]. Alexander documenta los patrones encontrados dentro de las ciudades, tales como vecindarios, áreas públicas y edificios especiales [24]. Las siguientes secciones discuten esta investigación en dos apartados: Red de transporte o tránsito principal y Vecindarios / Distritos

Red de tránsito principal

La red de tránsito principal consta de las carreteras o carreteras principales de una ciudad, las principales líneas ferroviarias y vías navegables. Esta red es un elemento importante para determinar la subestructura de una ciudad y es fundamental en la interpretación de la ciudad.

Desde una perspectiva visual,la red de transporte principal se percibe como el patrón más predominante y reconocible presente en la estructura de la ciudad. Esta teoría se puede reforzar con la investigación de Lynch. En la imagen de la ciudad, documenta una serie de razones de la importancia de la red de transporte principal y su relevancia en la percepción de la ciudad. En primer lugar, se introduce el concepto de Senderos, según Lynch "Los senderos son los canales por los que el observador se desplaza habitual, ocasional o potencialmente" [23]. En las ciudades, la mayor parte del transporte se realiza por carretera y la red de carreteras principales forma las principales arterias y canales de transporte. Estos caminos especifican el lugar y la ruta desde donde se observa la ciudad y, por lo tanto, dejan una impresión duradera en nuestra

imagen de la ciudad. Lynch no subestima la importancia de los caminos en la percepción de las ciudades, afirma que son los elementos predominantes en la imagen de la ciudad [23].

En segundo lugar, se describe el concepto más abstracto de Nodos. Los nodos se definen como los puntos estratégicos de una ciudad en los que puede entrar el observador y que son los focos intensivos hacia y desde los que se viaja [23]. Aunque no son específicos de la red de tránsito, se explica la relación entre los Nodos y la red: Pueden ser principalmente cruces, lugares de ruptura de transporte, cruce o convergencia de caminos [23]. Entonces, en relación con la red de tránsito principal, se puede fácilmente dibujar un paralelo entre los cruces como nodos y las carreteras o pistas como caminos. Ambos tienen un impacto significativo en la percepción y por lo tanto, juegan un papel importante en la determinación del carácter reconocible de un paisaje urbano.

Kelly y McCABE, aseguran que funcionalmente, la red de transporte principal es lo más importante para los usuarios de la ciudad [25]. Las ubicaciones y la estructura de esta red deben entenderse para navegar y moverse por la ciudad. Por eso al llegar a una nueva ciudad una de las primeras cosas que hacen los turistas es comprar un mapa . Dentro de un mapa de la ciudad, la red de carreteras principal se enfatizan para que parezca más evidente. La comprensión de la red de tránsito es esencial para la comprensión de una ciudad y es un factor clave en el reconocimiento de las características de la ciudad.

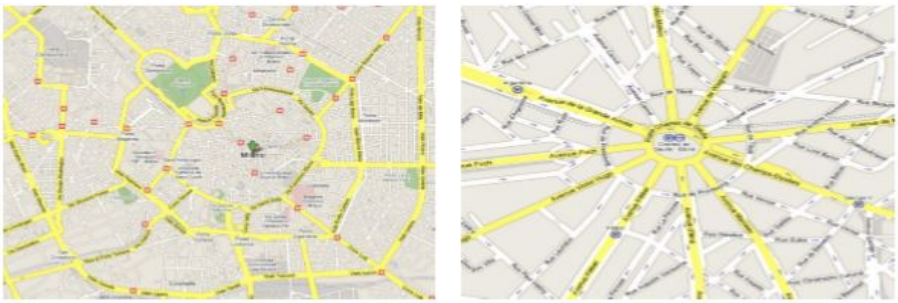

Figura 1.2. Dos redes de tránsito identificables:Milán(Izquierda),Paris(Derecha). Fuente:[26]

En la fig.1.2 Kelly y McCabe cuan visibles son los patrones en las ciudades y como esto forma una parte fundamental en el carácter de la ciudad.

Vecindarios

Los vecindarios, o distritos, son áreas de la ciudad que tienen un carácter identificable y un límite tangible. Cada ciudad está compuesta por distritos componentes, por ejemplo: el centro de la ciudad, el distrito financiero, áreas residenciales, áreas industriales, suburbios, etc. La investigación urbana ayuda a definir estas áreas y explicar las relaciones con los ocupantes de la ciudad. Lynch define los distritos como "secciones medianas a grandes de la ciudad, en las que el observador ingresa mentalmente 'dentro' y que son reconocibles por tener algún carácter común e identificativo" [23]. Alexander entrelaza la importancia de

los barrios y su relación con los ocupantes de la ciudad. En el patrón titulado Barrio Identificable, describe estas regiones y la relación con los ocupantes: "La gente necesita una unidad espacial identificable a la que pertenecer. Quieren poder identificar la parte de la ciudad donde viven como distinta de todas las demás " [24].

Estos componentes del paisaje urbano incluyen tanto fronteras naturales, como ríos y lagos, como también fronteras artificiales, como carreteras principales y vías férreas [24]. Lynch denomina los límites entre distritos como Bordes, que describe como rupturas lineales no consideradas como caminos [23]. Estos bordes pueden constar de elementos tales como costas, ríos, cortes de ferrocarril y carreteras principales. Aunque las vías férreas y las carreteras también son caminos, pueden considerarse bordes cuando actúan como una barrera al acceso de un distrito. Por ejemplo, considere una carretera principal que atraviesa una zona residencial, en este caso la carretera puede restringir el movimiento de peatones en el área local y ahora actúa más como un borde que como un camino. Alexander haciendo referencia al estudio de Appleyard-Lintell [27], define una métrica que indica que una carretera con más de 200 automóviles por hora deteriora la calidad del vecindario y forma una barrera para la circulación de peatones [24].

La identificación de los componentes que constituyen los distritos y sus límites asociados proporcionan una idea de la estructura de las ciudades. De la investigación urbana se desprende claramente que la red de transporte principal de la ciudad junto con las características naturales son clave para determinar los límites de estructuras más locales, como distritos o vecindarios. Con base en esta investigación, se puede concluir que la red de tránsito primaria es un aspecto importante del paisaje urbano y su efecto en la subestructura es amplio [23, 24]. Por lo tanto, se puede hacer una recomendación: para crear un sistema de generación de ciudad exitoso, se debe proporcionar un método accesible de control para este elemento influyente.

1.1.2. Generación Procedimental de Carreteras

Las redes de carreteras son un aspecto clave del carácter y la identidad de la ciudad. Las redes de carreteras son difíciles de generalizar porque son un componente entrelazado de un sistema complejo. Al visualizar las redes de carreteras desde un mapa o un plano de la ciudad, se pueden observar varios patrones. Estos patrones son clave para la generación de procedimientos, ya que codifican la estructura de la red de carreteras [22]. Las ciudades se pueden clasificar por los patrones de carreteras que contienen: las modernas, como Nueva York, están organizadas en un tablero de damas o patrón de trama, algunas ciudades europeas como París están estructuradas con un patrón radial o concéntrico más evidente [13]. Sin embargo, la mayoría de las ciudades contienen una serie de patrones diferentes que prevalecen en distintas regiones o vecindarios dentro de la ciudad.

La red de carreteras se divide en componentes más básicos, generalmente carreteras y sus intersecciones. Las carreteras están representadas por polilíneas y las intersecciones se describen como polígonos irregulares con modificaciones en las esquinas según diferentes requisitos [28]. Las intersecciones de carreteras son el

Patrón	Descripción	Ejemplo
Basic	Sin patrón superpuesto	
New York	Patrón rectangular	
Paris	Patrón radial	

Figura 1.3. Patrones. Fuente:[13]

punto de convergencia de las carreteras, la forma de un área de intersección depende en gran medida de las secciones de carreteras conectadas a su alrededor. Si bien el ángulo de intersección de las carreteras y su ancho se puede variar, la intersección invariablemente contiene dos tipos de bordes: (1) bordes que unen el área de intersección y la sección de la carretera (ver la línea azul sólida en la Fig.1.4) y (2) los bordes que conectan la sección de carretera y intersección (ver la línea roja discontinua en la figura 1.4).

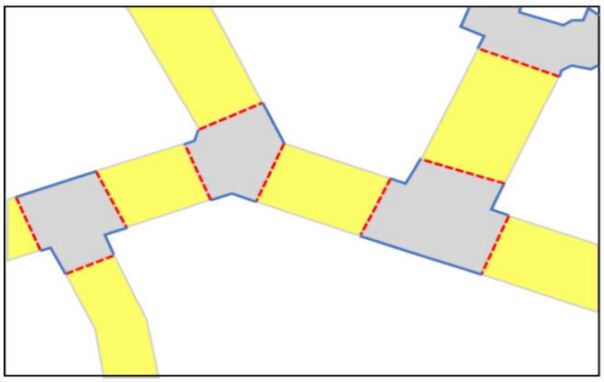

Figura 1.4. Intercepciones Fuente: [25]

Los bordes de tipo rojo se pueden establecer como un segmento de línea perpendicular al eje de la carretera, este tipo de borde determina la conectividad de la sección de la carretera y la intersección, hasta el modelo de red geométrica completa. El tipo de borde azul debería servir como una esquina de giro donde un arco podría adaptarse a esta parte. Este tipo de borde determina la continuidad de los bordes de la superficie entre la sección de la carretera y la intersección. Ambos tipos de aristas determinan directamente los efectos visuales del modelo geométrico generado [28].

Se han desarrollado muchos métodos para generar redes de carreteras de forma procedimental. Se han utilizado L-Systems (GG), simulaciones de agentes (CS) y campos tensoriales (CS) [29] para generar redes viales. Por ejemplo, se han utilizado L-Systems [13] para crear redes de carreteras con parámetros de control tales como densidad de población, patrones de carreteras, restricciones de elevación y restricciones locales, como aguas superficiales cercanas o (no) ubicación cerca de otras carreteras. Como ejemplo de simulación de agentes, uno para explorar el terreno y crear caminos, otro para conectar los caminos existentes entre sí en un árbol que se extiende por el terreno, se puede utilizar el método planteado por Thomas Lechner

[30]. Un método que combina la técnica GG y los diagramas de voronoi, como un lugar para las carreteras dentro de la ciudad y las carreteras entre ciudades, adicionalmente, se encuentra el que describió Jin Sun en su artículo [31]. Cabe destacar que cada uno de los métodos anteriores son los encargados principales en la generación de mallas 3D que conforman el modelo de la carretera.

1.2. Modelos de Representación Geométrica

1.2.1. Malla 3D

Es la representación geométrica de los cuerpos tridimensionales se utilizan modelos geométricos, para representar los objetos de modo que permita una visualización más exacta de diferentes elementos. Existen diversos objetos que conforman una escena virtual para proporcionar una mejor visualización de los elementos reales, estos crean los distintos tipos de modelos de representación geométrica, dentro de las más reconocidas están los modelos de mallas poligonales y los volumétricos [32].

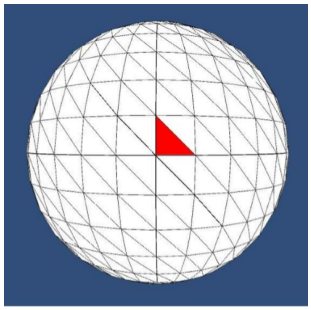

Figura 1.5. Malla de una esfera. Fuente:[33]

Mallas volumétricas

Los modelos de mallas volumétricas son ampliamente difundidos para la representación de órganos y reconstrucción 3D a partir de imágenes. Su principal objetivo es representar el interior de un objeto 3D para establecerlo como un cuerpo sólido. Están conformados por tetraedros, hexaedros o cubos por mencionar los más comunes. Lograr esta característica implica usar algoritmos de enmallado que permitan representar al cuerpo sólido. Los principales algoritmos para este proceso se dividen en algoritmo triángulo/tetraedro y algoritmo cuadrilátero/hexaedro [33].

El tipo de mallas tetrahédricas representa una malla volumétrica compuesta por muchos tetrahedros. Este tipo de malla se puede usar para realizar simulaciones y mantiene una considerable cantidad de información. Para cada tetraedro, las coordenadas de sus cuatro vértices se almacenan al igual que el número que indica el segmento al que pertenece el tetraedro, tal y como se obtuvo en el procedimiento de segmentación. Cuando se selecciona una malla, se muestra información sobre el número de sus vértices, aristas, caras y

tetrahedros [34].

Mallas poligonales

Los modelos poligonales son de gran uso en los gráficos por computadoras y en los programas de diseño 3D. La mayoría de los algoritmos en los motores gráficos están soportados con la manipulación de la información de triángulos en estructuras de datos, los que son considerados la base de las primitivas gráficas por su simplicidad y el soporte que brindan las herramientas actuales para su visualización [33].

Estas mallas suelen tener un gran número de nodos irregulares, y por lo general no hay una restricción fija sobre el número de elementos que confluyen en un único nodo. Otro inconveniente de este tipo de mallas es que apenas se puede controlar la forma y el tamaño de los elementos. No obstante, los algoritmos de triangulación (o tetrahedrización, para casos volumétricos) tradicionalmente empleados son capaces de descomponer las geometrías en triángulos que satisfacen criterios de calidad adecuados para el método de elementos finitos, como pueden ser restricciones de ángulo mínimo, de área, de altura, entre otros [35].

Después de planteadas las características de las principales representaciones gráficas de una malla se considera que el más adecuado para realizar el módulo es la malla poligonal. Debido a que la estructura a generar es hueca no necesita ser representada en su interior, además el coste computacional de la utilización de mallas poligonales es inferior y teniendo en cuenta que la estructura estará orientada a dispositivos móviles lo hace más factible.

1.2.2. Triangulación de Mallas

La triangulación es una división del área en un conjunto de triángulos que cumplen las siguientes condiciones: la unión de todos los triángulos es igual al polígono original, los vértices de los triángulos son vértices del polígono original y cualquier pareja de triángulos es disyunto o comparte únicamente un vértice o un lado, por lo que un algoritmo de triangulación es un método desarrollado para el cálculo de la triangulación de un polígono [33].

La triangulación es la técnica de generación de mallas no estructuradas más utilizada, cualquier geometría arbitraria se puede discretizar con menor dificultad con elementos triangulares que con cuadrangulares [35].Los algoritmos que se utilizan para la formación de la malla de triángulos irregular, se basan fundamentalmente en la triangulación de Delaunay, pues se trata de estructuras computacionales, que permiten la construcción de una triangulación óptima para la representación de mallas. Estos algoritmos, cumplen los condicionantes computacionales y geométricos, donde los triángulos formados son lo más regulares posibles y la longitud de los lados de los triángulos es mínima. La triangulación formada es única, dando lugar

a la red irregular de triángulos que aparentemente ofrece una imagen más fiel del terreno real, y que permite una interpolación coherente entre los valores de altitud de cada uno de los puntos o vértices [36].

Triangulación de Delaunay

La triangulación de Delaunay de un conjunto de puntos es una colección de aristas que satisfacen una propiedad de círculo vacío: para cada arista se puede encontrar un círculo que contiene los puntos finales de la arista pero que no contiene ningún otro punto. Es una de las triangulaciones más utilizadas por ser aplicable para la resolución de multitud de problemas, debido a sus propiedades geométricas y por contar con algoritmos bastante eficientes para su cálculo [36]. La triangulación de Delaunay es considerada una de las más uniformes, asegura que los ángulos del interior de los triángulos sean lo más grandes posibles, la longitud de sus lados es mínima y la triangulación formada es única [36].

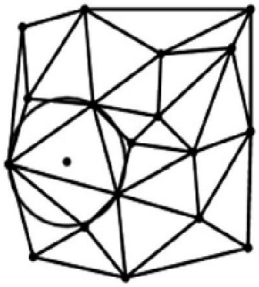

Figura 1.6. Resultado de una triangulación. Fuente: [37]

Según [37] la triangulación de Delaunay: Sea P = p1, p2,...,pn un conjunto de puntos en el plano [37], una triangulación de Delaunay de P cumplirá las siguientes propiedades:

- **Propiedad 1:** La triangulación de Delaunay tiene la propiedad de que la circunferencia circunscrita a cada triángulo no contiene a ningún otro punto de la triangulación.
- **Propiedad 2:** Dos puntos pi y pj pertenecientes a P forman un lado de la Triangulación de Delaunay de P, si y solamente si, existe un círculo que contiene a pi y pj en su circunferencia y no contiene en su interior ningún punto de P.

Con estas dos propiedades se puede caracterizar la triangulación de Delaunay de la siguiente manera: Sea P un conjunto de puntos en el plano y T una triangulación de P, T es una triangulación de Delaunay de P, si y solamente si, la circunferencia circunscrita de cualquier triángulo de T no contiene puntos de P. En la Fig.1.6 se puede observar el resultado de triangular una nube de puntos aplicando un algoritmo de Delaunay. A pesar de las bondades que posee este método de triangulación, cabe destacar que es utilizado para la triangulación de datos no estructurados, sin embargo el diseño del módulo esta previsto para que la entrada de datos a utilizar sea organizada, empleando los nodos del grafo para esta, por lo que realizar la triangulación

Delaunay ocasiona problemas en el ordenamiento de los triángulos. Se decide aplicar una triangulación propia descrita en el diseño de la propuesta del módulo, que se basa en la generación organizada de triángulos utilizando los nodos del grafo.

1.2.3. Grafos

El grafo es un término matemático utilizado para designar a un conjunto de puntos unidos entre sí por segmentos, que pueden representar un proceso o relación funcional de cualquier tipo, pero centrar su atención en las relaciones topológicas entre sus elementos [38].

Principales Clasificaciones

- **Grafos simples:** Un grafo es simple si a lo más existe una arista uniendo dos vértices cualesquiera. Esto es equivalente a decir que una arista cualquiera es la única que une dos vértices específicos. Un grafo que no es simple se denomina multigrafo.
- **Grafos conexos:** Un grafo es conexo si cada par de vértices está conectado por un camino; es decir, si para cualquier par de vértices(a, b), existe al menos un camino posible desde a hacia b. Un grafo es doblemente conexo si cada par de vértices está conectado por al menos dos caminos disjuntos; es decir, es conexo y no existe un vértice tal que al sacar el grafo resultante sea disconexo.
- **Grafo dirigido:** Un grafo dirigido conocido también como dígrafo consta de un conjunto de vértices y aristas donde cada arista se asocia de forma unidireccional a través de una flecha con otro. Las aristas dependiendo de su salida o ingreso, reciben la calificación de entrante o saliente, la condición común, es que siempre tienen un destino hacia un nodo.
- **Grafo no dirigido:** Los grafos no dirigidos son aquellos que constan un conjunto de vértices que están conectados a un conjunto de aristas de forma no direccional. Esto significa que una arista puede indistintamente recorrerse desde cualquiera de sus puntos y en cualquier dirección.
- **Grafos etiquetados:** Esta clasificación es denominada como grafos etiquetados o grafos dirigidos con pesos. Este tipo de grafos concentran aristas que pueden poseer información adicional donde se pueden reflejar nombres, costos, valores u otros datos. Estos grafos también son denominados como redes de actividad y el número asociado al arco, se le denomina factor de peso. Este es el que más comúnmente utilizamos para representar situaciones de la vida real.

Estructuras de datos en la representación de grafos

Existen diferentes formas de almacenar grafos en una computadora. La estructura de datos usada depende de las características del grafo y el algoritmo usado para manipularlo. Entre las estructuras más sencillas y usadas se encuentran las listas y matrices, aunque frecuentemente se usa una combinación de ambas. Las listas son preferidas en grafos dispersos porque tienen un eficiente uso de la memoria. Por otro lado, las matrices proveen acceso rápido, pero pueden consumir grandes cantidades de memoria.[38]

Estructura de lista

- Lista de incidencia: Las aristas se representan con un vector de pares (ordenados, si el grafo es dirigido), donde cada par representa una de las aristas.
- Lista de adyacencia: Cada vértice tiene una lista de vértices adyacentes a él. Esto causa redundancia en un grafo no dirigido (A existe en la lista de adyacencia de B y viceversa), pero las búsquedas son más rápidas, al costo de almacenamiento extra.

Estructuras matriciales

- Matriz de incidencia: El grafo está representado por una matriz de A (aristas) por V (vértices), donde [arista,vértice] contiene la información de la arista.
- Matriz de adyacencia: El grafo está representado por una matriz cuadrada M de tamaño, donde es el número de vértices. Si hay una arista entre un vértice x y un vértice y, entonces el elemento es 1, de lo contrario, es 0.

Para la realización del sistema de grafos en la creación de redes de carreteras se utilizó el tipo de grafo conexo, porque es necesario que para cada nodo en el mapa exista un camino de llegada. Ademas se empleara la estructura de matriz por el fácil acceso que se tiene a los datos en esta.

1.3. Soluciones existentes

La generación de ciudades se logra en 2 etapas, en cada una de ellas se aplica uno o más algoritmos para generar un componente constituyente de una ciudad. No hay un alcance predefinido para ninguna etapa y cada sistema tiene un enfoque único que dificulta las comparaciones directas. Sin embargo, el proceso de generación se puede dividir en dos etapas principales: generación de carreteras y generación de edificios. Debido a que el objetivo del análisis es más centrado en generación de carreteras, este será el tema en el que se hará énfasis en esta sección, donde se presentarán tres métodos viables para su generación. Se presenta una descripción general de cada sistema de generación de carreteras y se proporciona una idea del funcionamiento de las técnicas de procedimiento aplicadas.

Kelly y McCABE en su investigación proponen una serie propiedades a tener en cuenta a la hora de evaluar la generación, de estas propiedades solo se abarcarán 3 que se consideran las más importantes [25]:

- **Realismo:** Este determina que tan real es la ciudad modelada, además de cuántos detalles hay y qué tan cierto es el modelo generado con un modelo de ciudad real.
- **Control:** Este parámetro dicta en que medida puede influir el usuario en modelado de la ciudad, y define si este control es restrictivo o intuitivo.
- **Eficiencia:** Este parámetro recoge que tan eficiente es la creación del modelo.

Para la realización de la evaluación se utilizará el estudio de tres métodos diferentes que son de gran influencia en el estudio de la generación procedimental de contenido. Cada uno de estos ha sido de especial utilidad en el avance de esta tecnología.

1.3.1. CityEngine

Parish y Müller presentan una de las soluciones de generación de ciudades más completas, el CityEngine [13]. El CityEngine consta de un conjunto de componentes que incluyen la generación de carreteras, la construcción de edificios y la creación de caras de edificios que se unen para formar un flujo para la generación de ciudades. Los L-Systems se seleccionan como la técnica clave para la generación de procedimientos en CityEngine. Los sistemas Lindenmayer se han utilizado tradicionalmente para modelar fenómenos naturales, pero también son adecuados para la generación de ciudades debido a su naturaleza concisa, eficiencia computacional y propiedades de amplificación de datos [39].

Los L-Systems, como se discutió anteriormente, se han utilizado para modelar el desarrollo de plantas y estructuras de ramificación [39]. El CityEngine utiliza una forma extendida de L-Systems denominados L-Systems auto-sensibles para construir redes de carreteras de una manera que tenga en cuenta el crecimiento existente [13].

La entrada se toma en forma de mapas de imágenes 2D. Se requiere información geográfica sobre la elevación, la vegetación y los límites del agua, y también se pueden incluir mapas de imágenes socioestadísticas adicionales que especifiquen información como la densidad de población, el uso de la tierra, los patrones de las calles y las alturas máximas de los edificios [11]. Una aplicación de generación de red de carreteras, que se muestra en la Fig.1.8, se utiliza para gestionar la generación de carreteras y permite al usuario operativo especificar parámetros adicionales como el ángulo de suavizado de los bordes de la red de carreteras, el ancho de la carretera, entre otros.

La generación de carreteras se logra mediante el uso de dos conjuntos de reglas: los objetivos globales y las restricciones locales. Los tramos de carretera se trazan inicialmente de acuerdo con los objetivos globales que son similares a los objetivos que puede tener un diseñador de ciudades. Estos planes provisionales son luego refinados por las restricciones locales que reflejan las restricciones prácticas del mundo real y el estado de la red vial existente [13].

Objetivos globales:

- Existen dos tipos diferentes de carreteras: las carreteras principales conectan los centros de densidad de población que se pueden identificar a partir de un mapa de densidad de población en escala de grises que se proporciona en la entrada y las carreteras pequeñas conectan con la carretera principal más cercana.

- Las calles siguen un patrón geométrico superpuesto.
- Las calles siguen el camino de menor elevación.

Restricciones locales:

- Los tramos de carretera se podan para que quepan dentro de un área legal: los tramos de línea que se extienden hacia el agua se eliminan.
- Las carreteras se rotan para encajar dentro de un área legal: una carretera a la costa se curva alrededor de la costa como una carretera costera.
- Se permite que las carreteras crucen un área ilegal de cierta distancia: una carretera que se acerque a un tramo limitado de agua lo cruzará como un puente.
- Los tramos de carreteras se verifican para ver si se cruzan con carreteras existentes o si se encuentran a una cierta distancia de un cruce de carreteras existente: la Figura 1.7 muestra cómo se modifican los tramos de carretera propuestos para satisfacer las reglas de autoconfianza.

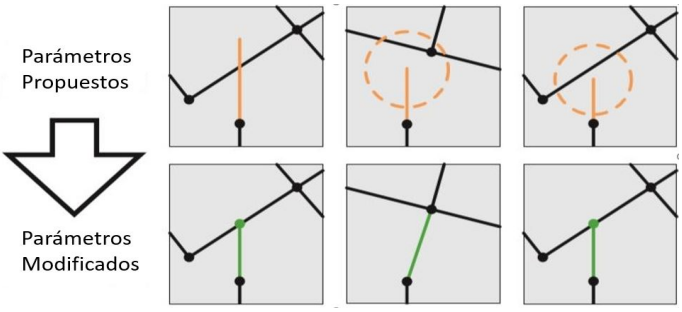

Figura 1.7. Ejemplo de corrección de parámetros. Fuente: [13]

Una vez que los valores de los parámetros están finalizados, el mapa de calles se puede renderizar o enviar a la siguiente sección de carreteras. Dado que el sistema solo crea tramos de carretera rectos, se necesita un método para suavizar las carreteras con una gran curvatura. La herramienta de creación de carreteras hace esto implementando un esquema de subdivisión simple basado en el algoritmo creado por Catmull y Clark [40].

Evaluación

- **Realismo:** El CityEngine puede crear una red de carreteras compleja y detallada, pero utiliza datos estadísticos reales que dificultan la evaluación de la capacidad de generación del sistema. Los bloques de la red de carreteras se dividen en lotes realistas y prácticos [13]. En general, se logra un buen equilibrio visual con un posicionamiento práctico.
- **Control:** No está claro cuánta interacción del usuario está permitida y no se documentan características interactivas. El sistema se controla escribiendo reglas específicas para cada región y, por lo tanto, se requerirían conocimientos y experiencia avanzados para utilizar el sistema. El control también está

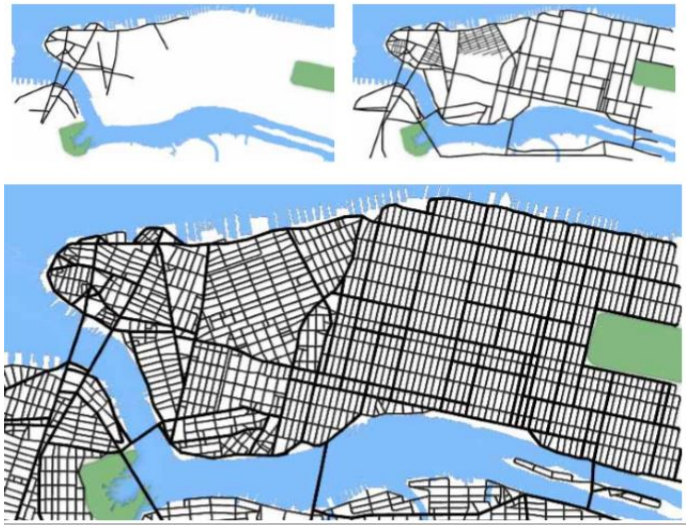

Figura 1.8. Interfaz de CityEngine mostrando una representación virtual de Manhattan luego de 100 pasos [11]

limitado con el uso de mapas de imágenes y no es adecuado para ser editado por un usuario novato [13].

- **Eficiencia:** La generación de redes de carreteras posee una elevada eficiencia en el uso de los recursos computacionales. La gran red de carreteras de la muestra de Manhattan en la Fig. 1.8 se crea en menos de 10 segundos.

1.3.2. CityBuilder

Lechner, Watson y Wilensky aplican una técnica basada en agentes para generar ciudades en su solución titulada CityBuilder [30]. El sistema está construido sobre la plataforma NetLogoTM, que es un entorno de modelado programable de múltiples agentes basado en el lenguaje de programación Logo y está diseñado para proporcionar a los usuarios una plataforma para explorar fenómenos emergentes. El sistema CityBuilder modela no solo la red de carreteras y los edificios, sino que también simula el crecimiento y desarrollo de la ciudad a lo largo del tiempo [30].

Las carreteras se crean a partir de segmentos de carretera que se ensamblan de acuerdo con un patrón de cuadrícula. Se permite la desviación del patrón y se puede especificar mediante un parámetro. Un valor de desviación de 0 resultará en una red de carreteras similar a una cuadrícula estrictamente uniforme, un valor de desviación cercano a 1 dará como resultado una red orgánica. La interconectividad de la red también se puede alterar a través de constantes que dictan la densidad de carreteras y la distancia entre las intersecciones de carreteras [30].

Lechner, Watson y Wilensky plantean que los agentes desarrollados para CityBuilder, interactúan direc-

tamente con su entorno y solo indirectamente entre sí, es decir, observando los efectos de otros agentes en el entorno [30]. Esto facilita la construcción de reglas, porque cada agente solo necesita preocuparse por el área local que lo rodea, en lugar de consultar el conjunto global de agentes activos en el mundo. Para facilitar el manejo del medio ambiente, este se representa como un conjunto rectangular de parches. Al igual que los agentes, cada parche también puede seguir comportamientos, pero están confinados a permanecer siempre en el mismo punto del espacio.

Se requiere la entrada en forma de un mapa de altura del terreno junto con un nivel de agua específico para determinar el área legal en la que se pueden colocar carreteras y edificios. Los parámetros adicionales como la densidad de la carretera, el espaciado de la cuadrícula y la desviación de la cuadrícula se pueden ajustar mediante controles deslizantes en la interfaz para alterar el comportamiento de los agentes. Además, los usuarios pueden especificar ciertos valores de parámetros para áreas específicas pintando en el mapa con un pincel similar al de una simple aplicación de pintura.

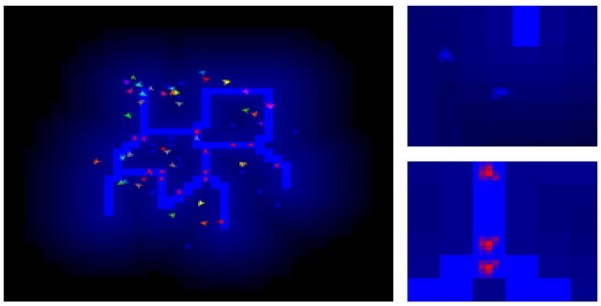

Figura 1.9. Generación de carreteras usando los agentes extensores (Img. izquierda arriba) y conectores (Img. izquierda abajo). Fuente: [30]

Los segmentos de carretera son creados por dos tipos de agentes: extensores y conectores [30]:

- **Extensores:** Un agente extensor recorre el terreno hasta que encuentra una ubicación que no cuenta con el servicio de la red de carreteras existente. Una vez que se ha descubierto esa área de tierra, sigue los valores de distancia hasta la red de carreteras y mantiene un registro de todos los parches por los que pasa durante su viaje. El terreno tiene cierta influencia en la decisión del agente, de modo que el agente trata de evitar tomar una ruta con un gran cambio de elevación. Una vez que el agente llega a la red, hace algunas pruebas de restricción sobre la conexión. Estas pruebas involucran problemas como la densidad de la carretera, la proximidad a las intersecciones existentes y la desviación de la posición inicial.
- **Conectores:** Un agente conector está obligado a deambular por la red de carreteras existente. Mientras recorre la red, muestrea un parche aleatorio en la carretera dentro de un radio determinado. Intentar llegar al parche seleccionado a través de la red de carreteras con una primera búsqueda amplia. Si no puede alcanzar su objetivo dentro del radio de muestreo original, o si debe alejarse demasiado de su camino, intentará construir un segmento de carretera entre su parche actual y el que no puede alcanzar.

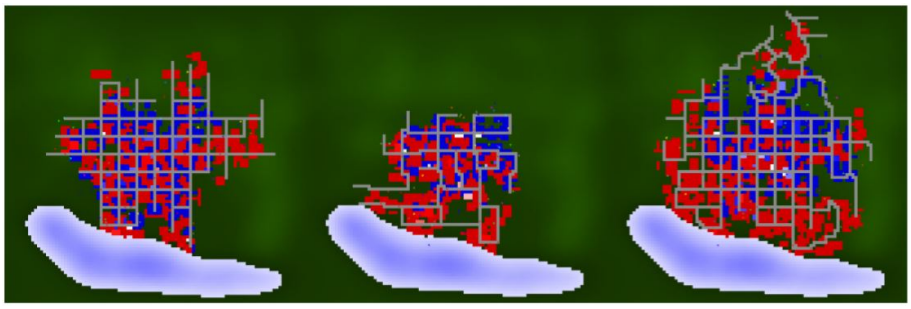

Figura 1.10. Generación de carreteras usando los agentes extensores (Ejemplo de salida de diferentes estructuras de la ciudad: 1) Cuadriculas, 2) Orgánico y 3) Mixto Cuadriculas y Orgánico). Fuente: [30]

Las redes de carreteras se pueden ver en evolución en tiempo real y los ejemplos que se muestran se crearon en un rango de 15 a 30 minutos. La Fig.1.11 c) muestra una de las principales fortalezas del sistema basado en agentes al combinar eficazmente los estilos de carreteras ráster y suburbanas.

Evaluación

- **Realismo:** La red de carreteras es realista y tiene la capacidad de realizar una transición eficaz entre los patrones de carreteras, especialmente entre las áreas urbanas centrales y las áreas suburbanas menos densas.
- **Control:** Una función innovadora está disponible en forma de herramienta de pintura que se puede utilizar para pintar valores de parámetros en el mapa. Los parámetros numéricos como la concentración de la carretera, la desviación y la escala se pueden especificar a través de una aplicación interactiva utilizando los diversos controles deslizantes y widgets de la GUI.
- **Eficiencia:** CityBuilder modela no solo la estructura de una ciudad sino también su evolución y, como resultado de la complejidad adicional, el algoritmo es computacionalmente intensivo y consume mucho tiempo. Se puede generar una ciudad de escala limitada similar a una aldea durante un período de aproximadamente 15 minutos.

1.3.3. Solución de campos tensoriales

En la propuesta de Chen y col. se introduce un marco de modelado intuitivo y flexible en el que un usuario puede crear una red de calles desde cero o modificar una red de calles existente. Esto se logra diseñando un campo tensorial subyacente y editando el gráfico que representa la red de calles. El marco es intuitivo porque utiliza campos tensoriales para guiar la generación de una red de calles. El marco es flexible porque permite al usuario combinar varias operaciones de modelado globales y locales, como trazos de pincel, suavizado, restricciones, ruido y campos de rotación. Los resultados obtenidos por Chen y col. mostrarán redes de calles y geometría urbana tridimensional de alta calidad visual [12].

El proceso de modelado de la herramienta está descrito en tres etapas principales [12]:

- **Primera etapa:** Para el inicio de la etapa el sistema debe recibir 4 mapas de datos principales:1) un mapa de agua con valor binario, 2) un mapa de parques y bosque con valor binario, 3) un mapa de altura y 4) un mapa de densidad de población. Una vez procesado los datos, el sistema permite que el usuario produzca un campo de tensión utilizando una variedad de operaciones de diseño, como combinar campos de base individuales, calcular campos de tensión desde los límites, utilizar la interfaz de pincelada y rotar el campo con ruido. Estas herramientas permiten al usuario refinar interactivamente el diseño.
- **Segunda etapa:** Es el paso de generación de las redes de carreteras. Las calles se calculan como hiperstreamlines del campo tensorial. Las redes de calles se modifican utilizando una jerarquía: carreteras principales y carreteras secundarias. Las carreteras principales suelen ser carreteras comerciales importantes y carreteras locales, y las carreteras secundarias suelen ser carreteras residenciales y callejones. Una red de calles se almacena como un gráfico G = (V, E) donde V es un conjunto de nodos y E es un conjunto de bordes. Los nodos con tres o más bordes incidentes son cruces. Los atributos de la vía, como el ancho de la vía, el tipo de vía, las marcas del pavimento y el tipo de carriles, se almacenan en los nodos y bordes.
- **Tercera etapa:** Es un módulo de generación de geometría que crea un árbol tridimensional y una geometría de construcción para obtener una ciudad completa.

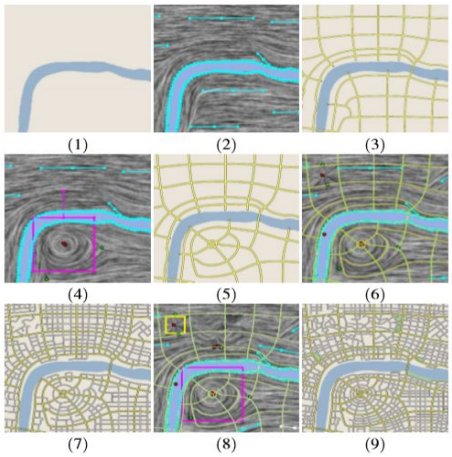

Figura 1.11. Proceso de generación de redes de carreteras en 9 pasos. Fuente: [12]

Una de las grandes potencialidades del sistema es la creación de patrones en las redes de carreteras bastante amplios, además de la combinación de estos patrones. Se le permite al usuario especificar patrones de red de calles deseados (por ejemplo, regular, radial, etc.) en las ubicaciones necesarias. Cada una de las restricciones especificadas se convierte en un campo de tensor de base definido para todo el dominio [12]. Estos campos luego se combinan usando funciones de base radial en descomposición, lo que permite mantener los patrones deseados en ubicaciones específicas. Para respetar las características en los mapas topográficos, también se generan campos de tensores de base que respetan los límites de las características,

como los límites de ríos y lagos.

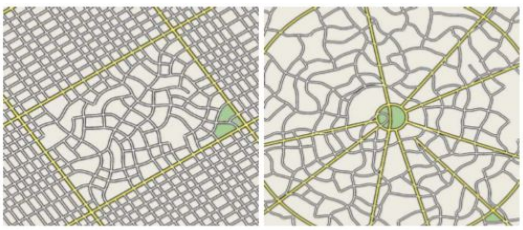

Figura 1.12. Ejemplos de combinación de patrones [12]

Evaluación

- **Realismo:** Las redes de carreteras creadas por el sistema poseen un alto nivel de realismo, la combinación de patrones de carreteras hacen que el diseño final sea una representación bastante fiel de estructuras de carreteras reales.
- **Control:** El usuario posee un alto control de la generación de carreteras, dándole la posibilidad de crear ambientes virtuales totalmente nuevos siempre teniendo en cuenta las restricciones del terreno.
- **Eficiencia:** Según los experimentos realizados por Chen y col., una ciudad con una complejidad razonable se puede modelar en cinco minutos, pero una ciudad con más dificultad estructural puede tomar entre 30 y 60 minutos es su creación

1.3.4. Resultados de la evaluación

Una vez analizadas las tres soluciones anteriores con respectivos algoritmos se obtuvo una idea clara sobre las fortalezas y limitantes de cada uno de ellos.

L-System: fortalezas y limitantes

- **Fortalezas:** Realiza un uso eficiente de los recursos computacionales, es de los tres algoritmos el más veloz para la generación de carreteras. Las redes de carreteras que se obtienen poseen un alto nivel de realismo, apegadas a las de ciudades reales.
- **Limitantes:** Se deben introducir muchos datos de entrada para obtener un modelo fiel. Solo genera segmentos rectos por lo que se debe utilizar algún método para suavizar las carreteras.

El algoritmo posee una eficiencia computacional excelente pero tiene la dificultad de que se deben entrar una cantidad elevada de datos geo-fisicos para lograr una buena representación, por lo que es usado generalmente en simulaciones. Se debe contar con personal especializado en la arquitectura de ciudades para hacer un buen uso de este algoritmo.

Simulación de agentes: fortalezas y limitantes

- **Fortalezas:** Genera una amplia variedad de patrones de ciudades, formando carreteras bastante realistas en su composición pero permitiendo libertad de diseño. Permite un mayor control de las modelaciones por parte de los desarrolladores gracias a que los agentes producen de forma independiente.
- **Limitantes:** Los tiempos para la generación de ciudades son bastante altos, pudiendo alcanzar hasta varias horas. [30]

A pesar de su alta tasa de tiempo es un algoritmo que permite mucha versatilidad en el diseño de las carreteras, lo que es de vital importancia para la creación de ambientes virtuales orgánicos y variados. Es un algoritmo práctico para el desarrollo de ambientes virtuales novedosos pero apegados a la realidad, lo que puede ser utilizado en la realización de videojuegos.

Campos tensoriales: fortalezas y limitantes

- **Fortalezas:** La representación de redes de carreteras posee el un grado de realismo muy alto, logra generar calles con altos niveles de curvatura. Puede generar una amplia variedad de patrones de ciudades e incluso la combinación de estos en el mismo modelo.
- **Limitantes:** Posee una dificultad de desarrollo elevada por el empleo, y necesita un amplio conocimiento sobre la arquitectura de ciudades. Muchos datos de entrada son necesarios para la generación de una red de carreteras

Debido a su alto realismo en la generación de carreteras, tiene gran aplicación en modelos arquitectónicos y ambientes de simulación de tránsito y densidad poblacional. Es el algoritmo con más versatilidad, pero también el más complejo debido a su rigurosidad matemática.

Para realizar el módulo se decidió no utilizar ninguno de los métodos mostrados anteriormente, ya que a pesar de ser muy eficaces, con gran muestra de realismo y veloces en ejecución, son utilizados para crear grandes entornos virtuales. Para el diseño del módulo se desarrollo un método propio con mucha mas sencillez y que cumple los objetivos propuestos, el cual es explicado en el diseño del módulo propuesto. Pero el estudio de estos tres métodos es fundamental por que otorga los conocimientos y herramientas necesarias para el diseño del módulo, como el recorrido del terreno usado en la simulación de agentes, que sirve de base para el recorrido de los nodos del grafo en donde evaluamos que tipo de carretera generar.

1.4. Tendencias y tecnologías

En esta sección se describen las diferentes tendencias y tecnologías asociadas al desarrollo del módulo como la metodología y las herramientas a utilizar.

1.4.1. Metodología de desarrollo

Una metodología de desarrollo no es más que un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. En un proyecto de desarrollo de software la metodología ayuda a definir quién debe hacer qué, cuándo y cómo debe hacerlo. La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende actividades a seguir para idear, implementar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado [41].

Hasta hace poco el proceso de desarrollo llevaba asociado un marcado énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada. Este esquema "tradicional" para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de ceremonia en el proceso. Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Ante las dificultades para utilizar metodologías tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo se resignan a prescindir del buen hacer de la ingeniería del software, asumiendo el riesgo que ello conlleva. En este escenario, las metodologías ágiles emergen como una posible respuesta para llenar ese vacío metodológico. Por estar especialmente orientadas para proyectos pequeños, las metodologías ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto [41].

XP (Extreme Programming)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en re-alimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios [42]. Se aplica mayormente en equipos de desarrollo pequeños o medianos, llevando una clara orientación tanto con los desarrolladores como con los clientes o usuarios finales [43]. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.En esta metodología los programadores trabajan en pares, desarrollando pruebas para cada función que quieran implementar [42].

La programación extrema consta de seis actividades estructurales: Exploración, Planeación, Diseño, Codificación, Pruebas y Muerte del Proyecto. Pero según Alvarez, De las Heras y Lasa se pueden unificar

en 4 fases manteniendo la funcionalidad de la metodología[44].

- **Fase de Planificación:** En esta fase se obtiene la información necesaria para empezar con el desarrollo del sistema, creando las historias de usuarios iniciales y el plan de iteración [44].
- **Fase de Diseño:** Esta fase guía la implementación de una historia de usuario; aquí se utilizan las tarjetas CRC y se hace uso del principio MS (mantenlo sencillo) para todo en lo que se esté trabajando [44].
- **Fase de Codificación:** En esta fase se desarrollan pruebas unitarias y se estimula la programación en parejas; a medida que las funcionalidades se van desarrollando se aplican estas pruebas, con el fin de retroalimentar el avance del producto [44].
- **Fase de Pruebas:** En esta fase se realizan las pruebas de aceptación, de manera que se verifique el funcionamiento del incremento de software que se ha realizado en la iteración [44].

Esta metodología utiliza 3 herramientas o artefactos fundamentales que nos ayudan a describir mejor las características del sistema[42]. Estos artefactos son:

- Historias de Usuario
- Tareas ingenieriles.
- Targetas CRC.

Las características fundamentales de esta metodología son [42]:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas: frecuentemente repetidas y automatizadas, incluyendo regresión de pruebas.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto.
- Frecuente interacción del equipo de programación con el cliente o usuario: se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.

Scrum

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle, define un marco para la gestión de proyectos que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos: primeramente el desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días, y segundo, las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración [45].

Cabe destacar que Scrum emplea un enfoque incremental que se adapta a los cambios que afecten al sistema, además de basarse en la experiencia para predecir y controlar riegos en el desarrollo [46]. A pesar de que Scrum surgió como un modelo para la gestión del trabajo de productos complejos, también se aplica en proyectos de software donde los requerimientos varían a medida que se desarrolla el sistema, donde la innovación y la adaptabilidad son parte fundamental del desarrollo y cuando se necesitan pequeñas liberaciones funcionales. Además, aquí se hace especial énfasis en el alineamiento entre el cliente y el equipo de desarrollo, de modo que el sistema se enriquezca de las aportaciones de todos los involucrados [47].

Podemos identificar en el patrón de ciclo de vida de un modelo de desarrollo ágil cinco fases [47]:

- Concepto
- Especulación
- Exploración
- Revisión
- Cierre

SCRUM plantea la realización de tres reuniones [47]:

- Planificación del Sprint
- Seguimiento del Sprint
- Revisión del Sprint

Sus principales ventajas son [47]:

- Se trabaja en iteraciones cortas, de alto enfoque y total transparencia.
- Acepta que el cambio es una constante universal y se adapta al desarrollo para integrar los cambios que son importantes.
- Se incentiva la creatividad de los desarrolladores haciendo que el equipo sea auto administrado.
- Permite producir software de una forma consistente, sostenida y competitiva.

Se escogió la metodología XP por la numerosa documentación que se encuentra sobre la misma, ademas de ser una de las más utilizadas en la actualidad y el centro posee mucha experiencia en la utilización de esta, ya que un gran número de los trabajos realizados en el la emplean. También se encuentra el hecho de que XP esta totalmente centrada proyectos pequeños y el mínimo de personas necesarias es 2 debido a la programación en pareja. Además esta metodología mantiene un mejor equilibrio entre la generación de artefactos ingenieriles y la libertad de los desarrolladores. En la investigación la trazaremos hasta la fase de diseño evidenciando la implementación de una historia de usuario y las tarjetas CRC, obviando las tareas ingenieriles ya que forman parte de la fase de prueba.

1.4.2. Herramientas y tecnologías

Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común, semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de software por ejemplo, en el flujo de procesos en la fabricación. Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene [48].

Herramienta de modelado

En la Universidad de las Ciencias Informáticas se ha estandarizado el Visual Paradigm que es una herramienta de Ingeniería de Software Asistida por Computación (CASE por sus siglas en inglés). La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Existe una alternativa libre y gratuita de este software, la versión Visual Paradigm UML 6.4 Community Edition (Community Edition, ya que existe la Enterprise, Professional, etc). Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos [49].

Motor gráfico e IDE

Como motor gráfico para el módulo se usará Unity, que es ampliamente empleado en el desarrollo de videojuegos multiplataforma creado por Unity Technologies, la biblioteca gráfica que aplica varía dependiendo del sistema operativo empleado, Direct3D (Windows), OpenGL (Mac y Linux). El éxito de Unity viene gracias al enfoque de este en las necesidades de los desarrolladores independientes que no poseen los recursos necesarios para hacer su propio motor o la adquisición monetaria para pagar las licencias de estos, por lo que hace a Unity el motor adecuado para ese tipo de personas [50].

Como IDE de desarrollo se usará Jetbrain Rider, la cual es un IDE .NET multiplataforma basado en la plataforma IntelliJ y ReSharper. Rider es compatible con proyectos basados en .NET Framework, el nuevo .NET Core multiplataforma y en Mono. Esto le permite desarrollar una amplia variedad de aplicaciones, incluidos servicios, bibliotecas y aplicaciones para escritorio .NET, juegos Unity, aplicaciones Xamarin y aplicaciones web ASP.NET y ASP.NET Core [51].

Herramienta para el control de versiones

Git, es un software de control de versiones diseñado por Linus Torvalds. Es encargado de la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo, es decir, a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración. Git fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente, Git les proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida [52].

1.5. Diseño del módulo propuesto

El desarrollo e implementación de este módulo esta concebido para formar parte de la herramienta de desarrollo de videojuegos Unity, y con este realizar la creación de sistemas de carreteras de una forma mucho más sencilla y ágil. Se pretende para el desarrollo del módulo es usar las bondades tecnológicas de la Generación Procedural de Contenido, para crear las distintas carreteras de una forma parametrizada y semi-automática. El usuario crea nodos y estos están conectados por las aristas, que serán los distintos tramos de carreteras, donde a cada nodo se le puede ajustar varios parámetros para obtener diversos resultados.

Para el desarrollo se tomó como base el sistema de generación procedural basado en el procesamiento de un grafo de nodos primitivos definido por el usuario. Este grafo determina donde la generación procedural tendrá lugar en el escenario. Los nodos definidos contarán con una serie de parámetros que especificarán como se genera la carretera de forma procedural.

1.5.1. Sistema de Grafo

Es una estructura de datos que contiene las listas de nodos y aristas, estos representan los tramos de carreteras y sus relaciones conformando de esta forma un grafo convexo. Esta estructura es la encargada de procesar los métodos de generación de carreteras haciendo uso de la información que proveen los nodos. Una matriz, es una forma compacta de representar el grafo teniendo en cuenta la estructura utilizada para almacenar los nodos y aristas, permitiendo tener diferentes propiedades como el ancho, los nodos siguientes y anteriores entre otras, que son datos definidos por el usuario y elementos a tener en cuenta para generar la carretera.

Nodos: Son la estructura base del sistema de grafos y los que poseen la información más relevante para tener en cuenta a la hora de generar contenidos. La disposición de esta información en Unity quedaría de la siguiente forma, donde Vector3 es una estructura de datos que representa la posición en el espacio de un objeto utilizando las coordenadas (x,y,z) [53].

```
public Vector3 Posición; //Contiene las cordenadas del Nodo
public Nodo[] nodosAnteriores;// Lista de nodos anteriores
public Nodo[] nodosSiguientes;// Lista de nodos Siguientes
public int peso;// valor que define el  ancho de la carretera
```

Todos los nodos ubicados quedaran almacenados en una lista de nodos, que el algoritmo debe recorrer hasta que todos hallan sido visitados.

1.5.2. Representación Geométrica

Las redes de carreteras en la realidad pueden ser variadas y complicadas,para la determinación y disposición de varios elementos de una carretera es necesario un trabajo de modelado detallado, incluyendo alineación vertical y horizontal, pendientes, distancia visual, anchos entre otras. Las herramientas anteriormente mencionadas en la Sección 1.2 remiten las normas de ingeniería civil a establecer los parámetros para determinar la geometría en su mayoría. Considerando la limitación de nuestro tiempo y recursos, este trabajo solo se centra en las características de la carretera que son suficientes para lograr una red razonable:

- **Tipo de carretera**
- **Ancho de la carretera**
- **Intersecciones**

Representación de la sección de carretera

Las carreteras son los elementos principales de la red de carreteras. Para el módulo se usan dos tipos principales de carreteras: **carreteras rectas y carreteras curvas**. Las carreteras rectas son creadas si dos nodos consecuentes poseen el mismo ángulo que el nodo anterior. Esto se puede obtener si los vectores normalizados de los nodos consecuentes poseen los mismos valores. Esto se obtiene de la resta n2 - n1, que da como resultado v1, luego se resta n3 - n2, se obtiene v2

```
Donde:
n1=> Vector del nodo actual
n2=> Vector del nodo siguiente a n1
n3=> Vector del nodo siguiente a n2
Por lo tanto:
v1= n2-n1
v2= n3-n2
Donde v1 y v2 son los vectores direccionales obtenidos de las restas
```

Al normalizar v1 y v2, si estos tienen el mismo valor la carretera a colocar es recta como se muestra en la Fig 1.13 a). En caso de que los vectores direccionales normalizados posean valores distintos se encontrara

ante una carretera curva como se muestra en la Fig 1.13 b). A través de los elementos aportados por el sistema de grafos se pudo definir el tipo de carretera que se debe generar, pero este sistema también define si estamos en presencia de una intersección, haciendo uso de los elemento **nodosAnteriores** y **nodosSiguientes**.

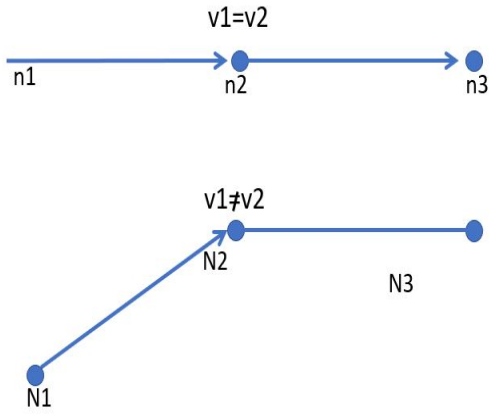

Figura 1.13. Representación geométrica de calle recta. Fuente: Elaboración Propia

```
Si nodosAnteriores > 1 entonces:
//Estamos en presencia de una intersección
Si nodosSiguientes > 1 entonces:
//Estamos en presencia de una intersección
Sino:
//Estamos en presencia de carreteras rectas o curvas
```

Detectados los tipos de carreteras existentes a través del sistema de grafos, se deben generar los vértices correspondientes para realizar la triangulación. Teniendo en cuenta que para cada nodo se generan 2 vértices perpendiculares al vector direccional formado entre el nodo actual y el siguiente (Fig 1.14).

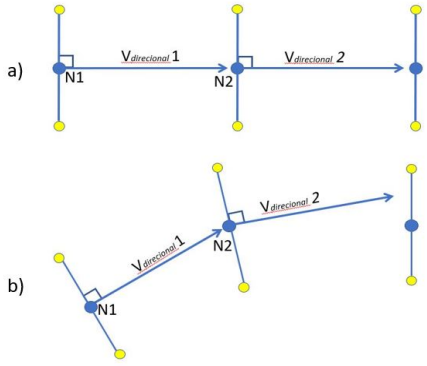

Figura 1.14. Colocación de los vértices con respecto a los nodos. Fuente: Elaboración Propia

```
Por lo tanto:
 Vector3[] verts = Cantidad de nodos*2 + Cantidad de intersecciones*2;
 Vector3 izq = vector direccional * peso; // Se coloca el vértice
 izquierdo usando el peso  el vector direccional
 Vector3 der = -Vector direccional * peso;// Se coloca el vértice
 derecho al lado contrario
 verts[vertIndex] = izq;// Se guarda el nodo izquierdo
 verts[vertIndex + 1] = der;// Se guarda el nodo derecho

 Donde:
 Vector3[] verts=> Es una lista con las posiciones de los vértices y su
  tamaño es la cantidad de vértices total del modelo
```

Con los vértices obtenidos solo resta aplicar una triangulación sobre estos para generar las mallas de la carretera. Se debe tener en cuenta que cada tramo de carretera esta conformado por dos nodos, por lo que según se explica anteriormente constará de 4 vértices, con los que se formaran los dos triángulos correspondientes a la malla de ese tramo de carretera, como se puede observar en la Fig 1.15. Cabe destacar que cada tramo de la carretera es representado como una arista del grafo que no es más que un par ordenado de nodos.

```
Por lo tanto:
int numRoad = Cantidad de nodos siguentes de cada nodo del grafo
int numTris = 2 * (numRoad + Cantidad de intersecciones)
Donde:
int numRoad = Número de tramos total del modelo
int numTris=> Es la cantidad de triángulos que posee el modelo
```

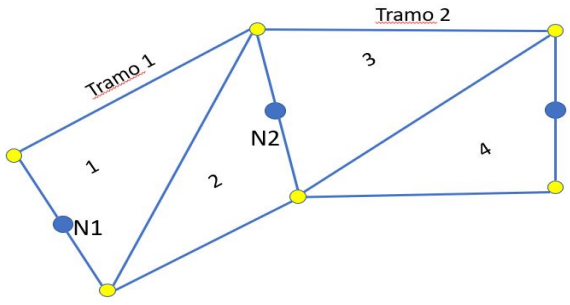

Figura 1.15. Colocación de los triángulos según los vértices generados. Fuente: Elaboración Propia

Intersección

La intersección de carreteras es el puntos donde se interceptan varias carreteras, es la clave para la flexibilidad de la red de carreteras. La intersección de carreteras se puede dividir en dos tipos: intersección de planos e intersección tridimensional. Como no se toma en cuenta la elevación, se enfocará principalmente en las intersecciones de planos y en particular, el método debería poder manejar la intersección de tridente, la intersección cuádruple y la intersección de anillo-isla en un plano. El método de modelado que divide la intersección se basa en crear un área poligonal en el centro de la intersección.

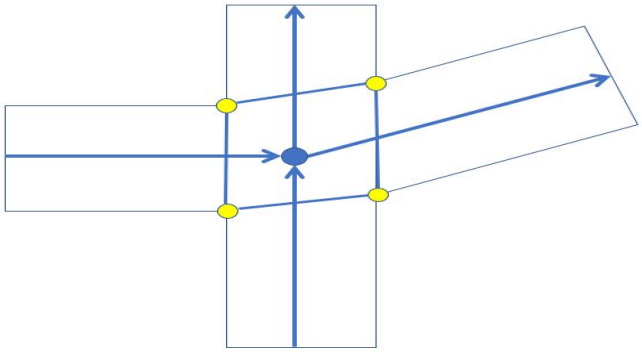

Figura 1.16. Representación de la intersección. Las flechas serían las carreteras, los puntos amarillos serían los vértices donde se interceptan estas carreteras formando el área poligonal, y el punto azul es el nodo de la intersección. Fuente: Elaboración Propia

Para lograr formar la intersección, el sistema obtiene la información de todas las carreteras conectadas en esta intersección, haciendo uso de la información aportada por los nodos en el sistema de grafo. Luego, calcula los nuevos puntos de intersección del borde lateral de cada camino, almacena los datos de estos puntos en una lista y ajusta el final de los tramos adyacentes. Después de finalizado el proceso anterior se crea un área poligonal en blanco en el centro de la intersección y se ajusta el área del polígono en función de los puntos de intersección calculados anteriormente.

Construyendo modelos 3D por scripts en Unity3D:

En Unity, los objetos se presentan como una combinación de mallas. Unity proporciona una clase Mesh que permite a los scripts generar o modificar la malla de objetos. Respecto a los objetos de la clase Mesh, Unity almacena los datos del modelo 3D en matrices [53] que son:

- **vértices:** la matriz de datos de vértice define la posición de la malla asociada con las coordenadas x, y, z de cada vértice;
- **triángulos:** la matriz de índice de vértice que define la forma en que serán conformados los triángulos.
- **normal:** la matriz de vectores normales, almacena la normal de cada vértice de la malla, y el tamaño corresponde a las coordenadas del vértice.

- **uv:** la matriz de coordenadas de textura, define la ubicación de cada punto de la imagen. Estos puntos están relacionados con el modelo 3D para determinar la ubicación del mapa de textura de la superficie.

```
Mesh mesh = new Mesh();// Inicializacion de la clase Mesh
mesh.vertices = verts;// Se le introduce la lista de vértices
mesh.triangles = tris;// Se le introduce la lista de triángulos
mesh.uv = uvs; tris;// Se le introduce la lista de cordenadas de
texturas

Donde:
mesh=> Es la variable que almacena la mallas una vez introducidas las
listas
```

Pseudo-Codigo

```
while (todos los nodos no estén visitados) do
        if(nodosSiguientes del nodo actual > 1)
                Definir como nodo de intersección
                Colocar vértices para intersección
        else if(nodosAnteriores del nodo actual > 1)
                Definir como nodo de intersección
                Colocar vértices para intersección
        else
                if(Nodo actual posee misma dirección que el siguiente)
                        Definir como carretera recta
                        Colocar vértices para carretera recta
                else
                        Definir como carretera curva
                        Colocar vértices para carretera curva
                end if
        end if
Siguiente nodo
end while
Calcular número de vértices
Calcular número de triángulos
Construir las mallas con apoyo de la clase Mesh de Unity
```

Con el diseño del módulo propuesto se logra una mejora de la visión final del componente y de esta forma, ya se tiene conformada la idea general sobre como funcionará el módulo. El sistema de grafo se convierte en la piedra angular de la solución, porque proporciona todos los datos necesarios para la generación de la estructura de las calles y describe las relaciones entre los nodos. Las redes de carreteras pueden variar considerablemente, diferentes variaciones darían lugar a diferentes tipos de carretera, con la descripción del método a implementar se da solución a los tres tipos de carreteras principales en este sistema. Haciendo uso de los vectores normales de los nodos se puede definir las carreteras con inclinación y rectas de una forma básica pero eficiente, y con la creación de un polígono ajustado a los vértices de intersección se satisface la creación de intersecciones. Además se realizó una descripción de los principales elementos a tener en cuenta en Unity para la creación de mallas para la implementación del módulo.

1.5.3. Historias de Usuarios

Representan una breve descripción del comportamiento del sistema, emplea terminología del cliente sin lenguaje técnico y se realiza una por cada característica principal del sistema. Se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación. Estas deben proporcionar sólo el detalle suficiente como para poder hacer razonable la estimación de cuánto tiempo requiere la implementación de la historia. Difieren de los casos de uso porque son escritas por el cliente, no por los programadores, empleando terminología del cliente.[42]

No.	Historia de usuario	Prioridad	Complejidad
1	Agregar nodo al mapa	Alta	Media
2	Eliminar nodo del mapa	Media	Media
3	Agregar los datos pertenecientes al nodo	Alta	Media
4	Generar vértices de las calles	Alta	Alta
5	Generar vértices de las intersecciones	Alta	Alta
6	Generar calles	Alta	Alta
7	Generar intersecciones	Alta	Alta
8	Exportar modelo	Alta	Media

Tabla 1.1. Historias de Usuario

Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarlas en unas semanas. A continuación la historia de usuario (Agregar los datos pertenecientes al nodo) para consultar las otras Anexo1:

En esta historia se muestra una descripción del requisito a implementar, así como la iteración en la que esta debe ser realizada. A continuación, en la Fig 1.16 se muestra como quedaría representado en el componente.

Historias de Usuario	
Numero:1	Usuario:Programador
Nombre: Agregar los datos pertenecientes al nodo	
Prioridad del negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 1	Iteración: 1
Programador Responsable: Adrian Preval González	
Descripción: Se debe poder agregar información acerca de los nodos	

Tabla 1.2. Historias de Usuario

Figura 1.17. Interfaz de entrada de datos. Fuente: Unity

1.5.4. Tarjetas de Contenido, Responsabilidad y Colaboración

La utilización de tarjetas CRC, es una técnica de diseño orientada a objetos. El objetivo de la misma es hacer, mediante tarjetas, un inventario de las clases que van a ser utilizadas para implementar el sistema y la forma en que van a interactuar, con ello se pretende facilitar el análisis y discusión de las mismas por parte de varios actores del equipo de proyecto, con el objetivo de que el diseño sea lo más simple posible,verificando las especificaciones del sistema.[42]

Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y colaboradores. A continuación la tarjeta CRC de la clase EditorPath, las otras tarjetas se encuentran en el Anexo2

Tarjeta CRC	
Clase: EditorPath	
Responsabilidad	Colaboración
Gestiona los métodos principales de generación de vías	PathCreator-EditorGui-Bezier

Tabla 1.3. Historias de Usuario

Conclusiones

En este trabajo se caracterizaron los elementos necesarios para el desarrollo de un editor en Unity que facilite la creación dinámica de estructuras de ciudades, dando cumplimiento al objetivo propuesto al inicio de la investigación, Adicionalmente, se concluye que:

- Para el caso particular del desarrollo de este módulo se creo un meto de generación procedural que se adecuara a las exigencias del proyecto, ya que los mas utilizados como el L-System y Agentes Simulados poseen una mayor robustez, y el módulo propuesto es un componente mas sencillo.
- Se creo un método propio que se adecuara a las propiedades del recorrido de grafos, además que fuera sencillo y fácil de implementar.
- En todos los casos, se recomienda el uso de grafos conexos para la estructura de datos subyacente, debido a que facilita naturalmente la representación de las relaciones entre los nodos y con esto, la representación de las intersecciones ente las calles.
- El diseño del módulo propuesto facilitaría la implementación en Unity de un editor simple y flexible, simplificado para videojuegos orientados a dispositivos móviles.

Recomendaciones

Se recomienda seguir con la investigación marcada en este trabajo, debido a que los métodos de generación procedural están en constante evolución y mejoramiento. Además se debe realizar el estudio de nuevas herramientas con nuevos enfoque para obtener aún más claridad en esta rama de los gráficos por computadoras.

Referencias bibliográficas

[1] *Origen de la pintura*. URL: https://curiosfera--historia-com.cdn.ampproject.org/v/s/curiosfera-historia.com/origen-de-la-pintura/.

[2] *Arte pictorico*. URL: https://documentosjalar-wordpress-com.cdn.ampproject.org/v/s/documentosjalar.wordpress.com/2016/11/07/el-arte-pictorico-y-su-relacion-con-la-historia-de-la-humanidad/.

[3] Raydan Carmelo. *Origen y expancion mundial de la fotografia*. Vol. 1. 2010.

[4] *Primera imagen digital*. URL: https://naukas.com/2010/11/10/la-primera-imagen-digital-de-la-historia-1957/.

[5] «Gimp,aplicaciones didacticas. La imagen digital». En: (2018).

[6] Jorge H Olivares. «Dispositivos de carga acoplada». En: (2005).

[7] *Crash Team Racing Nitro-Fueled Game | PS4 - PlayStation*. URL: https://www.playstation.com/en-us/games/crash-team-racing-nitro-fueled-ps4/.

[8] *Software de diseÃ±o3D, unpocodehistoria*. URL: https://laarquitecturadelobjeto-wordpress-com.cdn.ampproject.org/v/s/laarquitecturadelobjeto.wordpress.com/2013/12/09/software-de-diseno-3d-un-poco-de-historia/.

[9] Sebastian Meijer Mark Hendrick. «Procedural Content Generation for Game». En: (2012).

[10] Oliver Deussen y col. «Realistic modeling and rendering of plant ecosystems». En: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 1998, págs. 275-286.

[11] Pascal Müller y col. «Procedural modeling of buildings». En: *ACM SIGGRAPH 2006 Papers*. 2006, págs. 614-623.

[12] Guoning Chen y col. «Interactive procedural street modeling». En: *ACM SIGGRAPH 2008 papers*. 2008, págs. 1-10.

[13] Yoav IH Parish y Pascal Müller. «Procedural modeling cities». En: *Proceeding ofthe 28th annual conference on Computer graphics and interactive techniques*. 2001, págs. 301-308.

[14] Paola Cabrera Rodriguez Luis E Cubela Gonzalez. *La industria del videojuegos cubano, retos y oportunidades para su creacion*. Estudios de animacion ICAI, 2018.

[15] Pier Luca Lanz Julian Togelius Alex J. Champandard. «Procedural Content Generation: Goals, Challenges and Actionable Steps». En: *Artiï¬cialandComputationalIntelligenceinGames*. 1998, 61â75.

[16] David S Ebert y col. *Texturing and modeling: a procedural approach*. Morgan Kaufmann, 2003.

[17] Steve Dahlskog. «Patterns and procedural content generation in digital games: automatic level generation for digital games using game design patterns». Tesis doct. Malmö university, Faculty of Technology y Society, 2016.

[18] Mark Hendrikx y col. «Procedural content generation for games». En: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 9.1 (2013), págs. 1-22.

[19] Michael Nitsche. *Video game spaces: image, play, and structure in 3D worlds*. MIT Press, 2008.

[20] Leah Edelstein-Keshet. «Mathematical models in biology (classics in applied mathematics)». En: *Society for Industrial and Applied Mathematics, New York* (2005).

[21] DA Norman. «The design of everyday things Basic Books». En: *New York* (1988).

[22] K Dietrich, U Graf y M Rotach. *Strassenprojektierung*. Institut für Verkehrsplanung und Transporttechnik., 1975.

[23] Kevin Lynch. «Reconsidering the image of the city». En: *Cities of the Mind*. Springer, 1984, págs. 151-161.

[24] Christopher Alexander. *A pattern language: towns, buildings, construction*. Oxford university press, 1977.

[25] George Kelly y H McCABE. «An interactive system for procedural city generation». En: *Institute of Technology Blanchardstown* 25 (2008).

[26] George Kelly y Hugh McCabe. «A survey of procedural techniques for city generation». En: *ITB Journal* 14.3 (2006), págs. 342-351.

[27] Donald Appleyard, M Sue Gerson y Mark Lintell. *Liveable urban streets: Managing auto traffic in neighborhoods*. Vol. 76. 3. Department of Transportation, Federal HIghway Administration, 1976.

[28] XINGJIANG YU. «OSM-Based Automatic Road Network Geometry Generation in Unity». En: *DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING* (2019).

[29] Ruben M Smelik y col. «A survey of procedural methods for terrain modelling». En: *Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*. Vol. 2009. 2009, págs. 25-34.

[30] Thomas Lechner, Ben Watson y Uri Wilensky. «Procedural city modeling». En: *In 1st Midwestern Graphics Conference*. Citeseer. 2003.

[31] Jing Sun y col. «Template-based generation of road networks for virtual city modeling». En: *Proceedings of the ACM symposium on Virtual reality software and technology*. 2002, págs. 33-40.

[32] Josï¿½ Armando Aguilar. «Videojuegos». En: (2003).

[33] Borja Javier Herráez Concejo. «Segmentación y clasificación de mallas 3D». En: (2016).

[34] Juan Antonio Juanes Méndez y col. «Generación volumétrica de estructuras anatómicas, para su visualización espacial, que mejore la formación y el aprendizaje». En: (2011).

[35] D Javier Moreno Garrido. «Desarrollo y optimización de un generador de mallas superficiales y/o volumétricas para aplicaciones de simulación electromagnética». Tesis doct. Universidad de Alcalá, 2013.

[36] JOSÉ ENRIQUE PRIEGO DE LOS SANTOS y M ARIA JOAQUINA PORRES DE LA HAZA. «La triangulación de delaunay aplicada a los modelos digitales del terreno». En: *X Congreso del Grupo de Métodos Cuantitativos, Sistemas de Información Geográfica y Teledetección*. 2002.

[37] L.De Floriani, B. Falcidieno y C.Pienovi. «A Delaunay-Based Method for Surface Approximation». En: *Eurographics Conference Proceedings*. Ed. por P.J.W. ten Hagen. The Eurographics Association, 1983. DOI: 10.2312/eg.19831026.

[38] Guillermo Duran. *Teoria de Grafos*. Universidad de la republica, Montevideo, Uruguay, 2008.

[39] Aristid Lindenmayer. «Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs». En: *Journal of theoretical biology* 18.3 (1968), págs. 300-315.

[40] Edwin Catmull y James Clark. «Recursively generated B-spline surfaces on arbitrary topological meshes». En: *Computer-aided design* 10.6 (1978), págs. 350-355.

[41] Ian Sommerville. *Ingeniería del software*. Pearson educación, 2005.

[42] BR. NELDIN NOEL PÃREZ REYES BR. SINTYA MILENA MELÃNDEZ VALLADAREZ BR. MARIA ELIZABETH GAITAN. «METODOLOGIA ÃGIL DE DESARROLLO DE SOFTWARE PROGRAMACION EXTREMA». En: (2016).

[43] Marcos Klender Carrasco Gonzaga y col. «MetodologÃa HÃbrida De Desarrollo De Software Combinando Xp Y Scrum». En: *Mikarimin. Revista Científica Multidisciplinaria. e-ISSN 2528-7842* 5.2 (2019), págs. 109-116.

[44] Alonso Alvarez, Rafael De las Heras y Carmen Lasa. «Metodos agiles y Scrum». En: *Madrid: Anaya* (2012).

[45] Adriana Peralta. «Metodologia Scrum». En: (2003).

[46] Ken Schwaber y Jeff Sutherland. «The scrum guide-the definitive guide to scrum: The rules of the game». En: *SCRUM. org, Jul-2013* (2013).

[47] K Brito, D Sosa y K Hector. «Seleccion de Metodologias de desarrollo para aplicaciones web». En: *San Bernardino: Academica EspaÃ±ola* (2015).

[48] *UML, lenguaje de modelado*. URL: https://www.ionos.es/paginas-web/desarrollo-web/uml-lenguaje-unificado-de-modelado-orientado-a-objetos/.

[49] *Sitio web oficial de Visual Paradigm*. URL: https://www.visual-paradigm.com.

[50] *Sitio web oficial de Unity*. URL: https://www.unity.com.

[51] *Sitio web oficial de Jetbrain*. URL: https://www.jetbrain.com/es-es/rider/.

[52] *Sitio web oficial de Git*. URL: https://www.git-scm.com.

[53] *Sitio web oficial de Unity*. URL: https://docs.unity3d.com/.

Generado con LaTeX

Ap?ndices

APÉNDICE A

Desarrollo

A.1. Diseño de propuesta de módulo

Historia de usuario	
Numero:1	Usuario:Programador
Nombre: Agregar nodo al mapa	
Prioridad del negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 1	Iteración: 1
Programador Responsable: Adrian Preval González	
Descripción: Al presionar el click izquierdo en el escenario se debe crear un nodo	

Tabla A.1. Historias de Usuario

Historia de usuario	
Numero:2	Usuario:Programador
Nombre: Eliminar nodo del mapa	
Prioridad del negocio: Media	Riesgo en desarrollo: Media
Puntos estimados: 2	Iteración: 1
Programador Responsable: Adrian Preval González	
Descripción: Al presionar un botón de la interfaz se deben eliminar todos los nodos del mapa	

Tabla A.2. Historias de Usuario

A.2. Tarjetas CRC

Historia de usuario	
Numero:4	Usuario:Programador
Nombre: Generar vértices de las calles	
Prioridad del negocio: Alta	Riesgo en desarrollo: Alta
Puntos estimados: 3	Iteración: 2
Programador Responsable: Adrian Preval González	
Descripción: El módulo debe generar todos los vértices correspondientes a las calles	

Tabla A.3. Historias de Usuario

Historia de usuario	
Numero:5	Usuario:Programador
Nombre: Generar vértices de las intersecciones	
Prioridad del negocio: Alta	Riesgo en desarrollo: Alta
Puntos estimados: 3	Iteración: 2
Programador Responsable: Adrian Preval González	
Descripción: El módulo debe generar todos los vértices correspondiente a las intersecciones	

Tabla A.4. Historias de Usuario

Historia de usuario	
Numero:6	Usuario:Programador
Nombre: Generar las calles	
Prioridad del negocio: Alta	Riesgo en desarrollo: Alta
Puntos estimados: 3	Iteración: 2
Programador Responsable: Adrian Preval González	
Descripción: El módulo debe generar todos las carreteras correspondientes	

Tabla A.5. Historias de Usuario

Historia de usuario	
Numero:7	Usuario:Programador
Nombre: Generar las intersecciones	
Prioridad del negocio: Alta	Riesgo en desarrollo: Alta
Puntos estimados: 3	Iteración: 2
Programador Responsable: Adrian Preval González	
Descripción: El módulo debe generar todos las intersecciones correspondientes	

Tabla A.6. Historias de Usuario

Historia de usuario	
Numero:6	Usuario:Programador
Nombre: Exportar modelo	
Prioridad del negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 1	Iteración: 3
Programador Responsable: Adrian Preval González	
Descripción: Se debe presionar un botón y exportar el modelo	

Tabla A.7. Historias de Usuario

Tarjeta CRC	
Clase: PathCreator	
Responsabilidad	Colaboración
Es el que se encarga de la creación de los nodos	EditorGui-NodeData

Tabla A.8. Historias de Usuario

Tarjeta CRC	
Clase: EditorGui	
Responsabilidad	Colaboración
Es la clase que se encarga de gestionar las opciones del módulo	RoadMesh-PathCreator

Tabla A.9. Historias de Usuario

Tarjeta CRC	
Clase: RoadMesh	
Responsabilidad	Colaboración
Es la clase que se encarga de generar las mallas y vértices	EditorGui

Tabla A.10. Historias de Usuario

Tarjeta CRC	
Clase: NodeData	
Responsabilidad	Colaboración
Es la clase que almacena las métricas de los nodos.	PathCreator

Tabla A.11. Historias de Usuario