UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA Y CIENCIAS
DE LA COMPUTACIÓN

PROCEDURAL CONTENT GENERATION PARA MODIFICAR LA EXPERIENCIA DE JUEGO

POR

DIEGO DOMÍNGUEZ RIVERA

MEMORIA PRESENTADA PARA LA OBTENCIÓN DEL TÍTULO DE

INGENIERO CIVIL INFORMÁTICO

PATROCINANTE

JULIO ERASMO GODOY DEL CAMPO

Enero de 2020

Concepción, Chile

Índice

Índice de imágenes

Resumen

Los videojuegos son parte importante en la vida moderna, principalmente como modo de entretención. De manera similar a la industria del cine, los videojuegos son piezas de software bastante costosas de producir, lo que crea la necesidad de acelerar procesos de desarrollo. Para ayudar en esta situación emerge el concepto de Procedural Content Generation, para producir distintos assets de videojuegos de manera algorítmica, disminuyendo la intervención humana en los procesos de desarrollo. Este concepto es bastante amplio y ha crecido conforme a los avances científicos y tecnológicos, encontrándose con otras áreas como la inteligencia artificial y la información geográfica produciendo diferentes productos. En esta memoria de título se propone un acercamiento a la generación de escenarios de forma procedimental con ayuda de datos de ubicación del jugador, datos geográficos y métodos de PCG para producir entornos más familiares para los usuarios y enriquecer la experiencia de juego. Para validar los datos se realizó una encuesta para evaluar los resultados de los escenarios generados, produciendo respuestas mayoritariamente positivas en cuanto a la experiencia de juego y la apelación a la familiaridad y nostalgia que producen escenarios basados en ubicaciones reales y conocidas, demostrando, además, potencialidades del proyecto para ser aplicado en otras áreas.

1. Introducción

La industria de los videojuegos ha tenido un crecimiento sostenido en los últimos años. En el año 2016, esta industria generó US$ 91 mil millones de dólares alrededor del mundo e involucra a diversas disciplinas de trabajo durante los procesos de desarrollo [1]. Su alza ha llevado a que en 2018, en Estados Unidos, la industria supere la recaudación del cine, considerando todas las plataformas (PC, smartphones, consolas, etc) [2].

Al igual que el cine, los videojuegos son un medio de entretenimiento costoso de producir. La industria ha crecido de la mano con los avances tecnológicos en hardware y software, lo que provoca un aumento en la complejidad de desarrollo, es decir, más tiempo y dinero invertido en la producción. Considerando que existen diversos estudios desarrolladores con distintos niveles de presupuesto (AAA, indie), se han invertido recursos en la búsqueda de soluciones para disminuir tanto los tiempos de desarrollo como los costos asociados. Esto, ha llevado al surgimiento de métodos que ayudan a la producción de recursos como texturas, sonido, objetos, etc. de forma algorítmica. Estos métodos se engloban bajo el término **Procedural Content Generation** [3] y permiten a los desarrolladores ahorrar tiempo, espacio y dinero para la creación de activos para videojuegos (de ahí su uso en la década de los 80 y 90 por las limitaciones del hardware). Actualmente, su uso va desde otorgar aleatoriedad a distintos componentes de los videojuegos [3], hasta complejas herramientas de apoyo al desarrollo para la creación de escenarios, animaciones y recursos en general.

1.1 Descripción del problema

En la actualidad, se puede observar que existen videojuegos en el mercado que pueden volverse un tanto monótonos y repetitivos a medida que el usuario dedica tiempo en el juego, ya que la selección de mapas es muy limitada e incluso única. Casos como estos se pueden encontrar en videojuegos populares como Apex Legends, Player Unknown: Battlegrounds (en su primer año de vida), que cuentan con un solo escenario extenso que dan a pensar todo el tiempo de desarrollo que se necesitó para poder realizar diseños, balance, decorados, texturizados, creación de puntos de interés y distribución de elementos. La falta de variedad en el catálogo de escenarios en videojuego o la presencia de sistemas de votación que producen la misma selección de escenarios una y otra vez, puede llevar a jugadores a perder el interés en el peor de los casos [4].

Como una opinión más personal, que también comparten otras personas [5], podemos considerar los casos en que los videojuegos han realizado una reimaginación de sitios reales, pero de una forma muy alejada a la realidad. Podemos citar el caso del videojuego **Battlefield Bad Company 2**, un shooter en primera cuyo enfoque primario es la experiencia multiplayer. En el modo multiplayer se podían escoger varios mapas inspirados en Chile y sus alrededores (*"Atacama Desert"*, *"Valparaíso"*, *"Arica Harbor"*, *"Isla Inocentes"*, entre otros) [6], ya que en esos lugares se manejaba la historia principal del juego. También se puede mencionar el caso del videojuego **Hitman: Blood Money**, en la misión *"A Vintage Gear"*, donde el juego nos lleva a una reimaginación de estilo tropical de una hacienda en el Valle de Colchagua, en la sexta región [7]. Si se realiza una comparación visual entre la realidad y el juego, rápidamente se concluye que lo único similar entre lo real y el juego es el nombre y en algunos casos, el clima. Por otro lado, tenemos un fragmento de una reseña realizada al videojuego chileno Abyss Odyssey, en donde la persona expresa su emoción por ver representado en el juego a parte de la ciudad de Santiago del siglo XIX y símbolos patrios: cita: *"Es un deleite visual, me gusta ya que como chileno, hay tantos detalles que me hace sentir orgulloso como el Moai, el Copihue que es la flor nacional, escudos militares y escenarios del antiguo Santiago, el diseño de los personajes y su historia, es muy cultural. A mi (me) gustó, quedó totalmente impresionante"* [5]. Así, aún cuando la representación de los lugares no se asemeja a la realidad, se produce cierta emoción como jugador al saber que un videojuego tenga mapas jugables inspirados en nuestro país, de la misma manera que cuando un documental extranjero muestra lugares de Chile.

A partir de lo expuesto, Procedural Content Generation es un concepto ampliamente utilizado en el desarrollo de videojuegos que aceleran los tiempos de producción y costos de desarrollo, además de facilitar la innovación y creatividad de level designers. Además, se puede fijar como hipótesis que la nostalgia y familiaridad que puedan tener los jugadores con lugares que hayan visitado son conceptos que pueden ser aprovechados al desarrollar escenarios para videojuegos con el fin de brindar una mejor experiencia.

1.3 Solución Propuesta

Se pretende utilizar los datos de la ubicación del jugador y hacer una aproximación para generar un escenario que tome la forma de alguna ciudad, comuna, provincia o región próxima utilizando datos espaciales reales y poblar este escenario con elementos de juego utilizando algoritmos propios de procedural content generation según datos geográficos, con

el fin de simular los ambientes reales de cada ubicación tratando de apelar a la nostalgia, aprovechando el nivel de familiaridad de los jugadores con los lugares.

1.4 Objetivo General

Diseñar e implementar un generador de escenarios que haga uso de datos espaciales, herramientas de geolocalización y métodos de procedural content generation para variar y enriquecer la experiencia de juego.

1.5 Objetivos Específicos

- Integración de geolocalización dentro de un videojuego para la generación de contenido.
- Implementación de algoritmos y métodos comunes dentro del área de procedural content generation para la distribución de elementos de juego dentro del escenario generado.
- Demostrar de manera cuantitativa las diferencias entre las experiencias de los usuarios al probar elementos de juego generados algorítmicamente.

2. Discusión Bibliográfica

En esta sección se discuten conceptos y trabajos relacionados que hayan utilizado técnicas de procedural generation y/o geolocalización, que pueden servir de ayuda para cumplir los objetivos planteados. Se presentan secciones de contenido en base al libro ***Procedural Content Generation in Games*** [8] y algunas de las soluciones disponibles en el mercado relacionadas a generación de escenarios, terrenos y utilización de información geográfica.

2.1 Generación de Terrenos en videojuegos

2.1.1 Nivel Conceptual

En el libro Procedural Generation in Games, encontramos la temática de la generación de terrenos y como distintos conceptos ayudan en su producción [8]. Uno de los puntos mencionados corresponde a los heightmaps, que corresponde a un array bidimensional de números reales a partir del cero, en el cual el valor de cada celda de la matriz corresponde a una altura desde el punto base. Aquí es donde alguna técnica para generar ruido (noise) ayuda a llenar estos valores, pero no cualquier ruido es útil para generar un terreno. Un

ruido que genere números aleatorios de forma independiente tiene una alta probabilidad de generar muchos picos en el terreno, es por esto que existen técnicas de generación de ruido basadas en gradientes, como **Perlin noise** [46], para producir ruido controlado.

El ruido, por sí solo no produce resultados muy similares a un terreno real. Si se observa desde baja a gran escala, la naturaleza tiende a repetir ciertos patrones, es por esto que si tenemos una vista en la que un paisaje plano se levanta a cadenas montañosas, dentro de esas cadenas montañosas también podemos encontrar valles menores y picos aún más inclinados. Esta situación es la base de los **Fractals** (fractales) y es posible generar terrenos con estas propiedades, al igual que producir otros tipos de contenido como vegetación, pasto, hojas árboles, etc.

Estas técnicas, entre muchas otras, son aplicadas a la generación de terrenos, incluyendo técnicas basadas en agentes, algoritmos genéticos, algoritmos de búsqueda, etc.

2.1.2 Herramientas de apoyo

Si consideramos la generación de contenido durante etapas de producción, en la industria de los videojuegos podemos encontrar diferentes herramientas que usan procedural generation para acelerar la producción de entornos jugables, ya sea como software independiente, o como extensiones al editor del motor gráfico a través de paquetes. De este tipo de herramientas podemos encontrar **World Machine** [9], **World Creator** [10], **Gaia** [11], **Terragen** [12], etc. La mayoría son de pago, funcionan como herramientas de apoyo a los Level Designer y tienen la finalidad de producir terrenos extensos a través del uso de algoritmos y estructuras de datos de uso común en herramientas de generación de terrenos procedimentales, de los cuales se puede mencionar estructuras como Heightmaps, Flow Maps, Slope Maps, Abstract Maps, máscaras, etc. y algoritmos como Perlin Noise, Fractal Noise, Clustering, Fluid Simulation, etc.

2.1.3 Utilización de información geográfica

A. Videojuegos populares

Los acercamientos más comunes a la producción de videojuegos utilizando datos geográficos se dan en la industria para móviles. Dadas las características de estos dispositivos, los smartphones tienen la capacidad de dar una localización del jugador más o menos precisa en tiempo real gracias a la utilización del sensor GPS

integrado y las bondades de las tecnologías de realidad aumentada (**AR**). Podemos tomar como ejemplo el videojuego móvil **Pokemon GO** o **Harry Potter: Wizards Unite** (creados por el mismo estudio, Niantic Labs) [13] que están basados en una premisa similar al utilizar una versión simplificada del mundo real como escenario donde se desarrolla el juego (figura 1).

Figura 1. Mapa Pokemon Go.

B. Location as a Service (LaaS)

Location as a Service es un concepto incorpora pilares fundamentales de Cloud Computing: Infraestructura, Plataforma y Software como servicio, brindando datos localizados a los clientes a través del uso de APIs [14][15]. El mejor ejemplo actual de este concepto es **Mapbox**. Mapbox es una plataforma de datos de ubicación para aplicaciones web y móviles, que provee todo lo necesario para agregar mapas, búsqueda y navegación a las aplicaciones que desee [16]. Específicamente, Mapbox provee de un SDK (Software Development Kit) para Unity, llamado **Mapbox Maps SDK for Unity** que es una colección de herramientas para crear aplicaciones Unity utilizando datos reales, interactuando con distintos servicios de Mapbox [17]. Este es uno de los acercamientos más fuertes en cuanto a juegos con geolocalización se refiere, ya que propone varios acercamientos de producir juegos utilizando datos de mapas como base para crear escenarios. Sin embargo, carece de integración con otros sistemas de Unity, como por ejemplo el sistema de terreno, pues utiliza un entorno propio dentro del motor que depende de una herramienta externa llamada **Mapbox Studio** para personalizar la forma en que se presenta la información geográfica en el juego, lo que no deja mucho margen para aprovechar las

optimizaciones de Unity para crear entornos complejos y muy detallados, lo que da a entender que su enfoque es a entornos móviles.

3. Método Desarrollado

Teniendo en cuenta la solución propuesta, con el fin de generar un prototipo que difiera de las soluciones ya existentes se propone una estructura de un generador de escenarios que utilice la ubicación del jugador, datos espaciales y aplicación de algoritmos de PCG para generar entornos jugables que sean familiares para el jugador.

Estructura del prototipo:

- Desarrollado utilizando el motor gráfico **Unity** [18].
- Sea capaz de:
    - Obtener la ubicación del jugador.
    - Uso de datos espaciales del territorio nacional en forma de heightmap para producir la geometría del escenario jugable.
    - Tener un sistema simulando zonas naturales para texturizado y decorado de cada región.
    - Recuperar datos geográficos para poblar el escenario con puntos de interés y elementos de cada región de forma más o menos precisa.
    - Tener un sistema de dia y noche coordinado al tiempo real.

En la siguiente sección se detallan las etapas y procedimientos que se incluyen en el proceso de generación de escenarios, como la obtención de datos, edición de heightmaps, texturizado y el ciclo día noche.

Además, con el propósito de validar el cumplimiento de los objetivos planteados, se realizará una etapa de testing del prototipo con jugadores, que incluye una encuesta que busca evaluar distintos aspectos del prototipo y cómo estos influyen en la experiencia general de juego.

4. Ejecución de Generación de Escenarios

Para la generación de los escenarios, la idea principal es poder generarlos a través de procedimientos, utilizando datos de ubicación, datos espaciales y algoritmos sobre las herramientas que dispone el motor Unity, creando un sistema que se pueda diferenciar respecto a las otras soluciones que se presentan en el mercado.

Si se considera las capacidades del motor gráfico utilizado, se puede encontrar la herramienta para la producción de terrenos, llamada **Terrain Engine** [19]. Este sistema está creado pensando en Level Designers para crear extensos paisajes 3D con distintas herramientas que brindan flexibilidad para incluir distintas texturas, objetos, árboles, pasto, niveles de detalle, etc. En este entorno, **Terrain** corresponde a un plano (Mesh), cuya forma va dependiendo de cómo el usuario "pinta" sobre el plano al utilizar las herramientas para modificar la altura del terreno y, además, permite importar y exportar **Terrain Heightmaps** [35]. Los heightmaps son un tipo de imagen ráster utilizado comúnmente como un modelo de elevación digital (DEM) en software GIS. Cada píxel guarda valores que representan datos de elevación para poder mostrarlo como un gráfico 3D. Este tipo de imagen tiene varios usos, como por ejemplo bump mapping (3D Rendering), para añadir sombreado a un material, o para la representación de terrenos, donde el el heightmap es convertido a un mesh 3D [20] (ver figura 2).

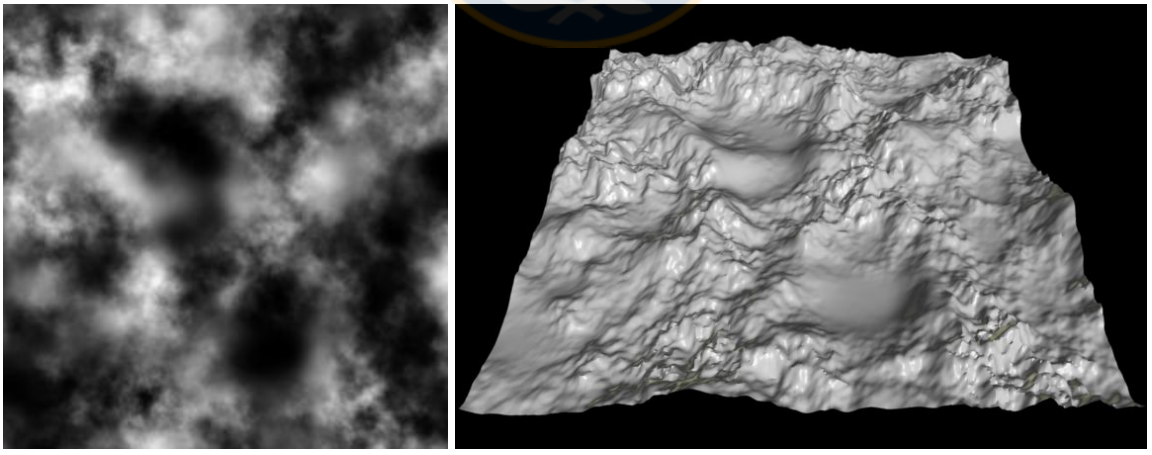

*Figura 2. Heightmap creado utilizando software **Terragen** (izquierda). Mesh 3D del mismo Heightmap renderizado en **Anim8or** (derecha).*

Ahora, lo que importa es lo que hay detrás del motor de terreno, las estructuras de datos y métodos que soporta para así poder manipularlo mediante código. Es aquí donde entra en juego la clase **Terrain** [21], ya que permite realizar esta tarea. En esta clase, con el objeto

que principalmente se trabaja para manipular el terreno es el objeto **TerrainData** [22]. Este objeto es el más importante de la clase, ya que almacena todos los datos, objetos, heightmap, texturas, vegetación, detalles, etc. que se colocan sobre el terreno, lo que sumado a las capacidades de las librerías integradas del motor, permite cargar estos datos en tiempo de ejecución.

En un principio, el punto de partida de generación de escenarios es establecer la ubicación del jugador. Para esto, se averiguó sobre distintas formas de obtener esa información y, en el ámbito de software de escritorio, el acercamiento más factible es utilizar localización por IP, que no es perfecto, pero es suficientemente preciso para determinar la cuidad en dónde se encuentra el usuario y, por ende, la región.

Luego, para poder definir la forma del terreno en cada caso, se tiene que disponer de los heightmap que correspondan a cada región del país. Para llegar a ese punto, es necesario tener heightmap que cumplan con distintos criterios para poder ser importados, por lo que se necesita utilizar dos herramientas software para lograr este fin: QGIS y GIMP. Por un lado, **QGIS** es un software GIS (Geographic Information System) gratis y de código abierto que es capaz de manejar distintos conjuntos de datos de tipo vectorial y ráster y dispone de una variada gama de funcionalidades y complementos para visualizar, gestionar, editar y analizar datos, y diseñar mapas imprimibles [23]. Con este software se pueden manipular datos geográficos del país obtenidos de sitios gubernamentales como la **IDE Chile** y la **SRTM** de la **NASA** que en conjunto, utilizando las capacidades de QGIS, se puede producir parte importante de los recursos necesarios para producir los heightmaps. Por el otro lado, solo utilizando QGIS no es posible producir los heightmap que se necesitan, por lo que se necesita de edición de imágenes para poder lograr que los heightmap cumplan con las exigencias de Unity. **GIMP** [24] es un software de edición de imágenes muy similar a Adobe Photoshop que permitirá realizar las ediciones necesarias para cumplir con los objetivos.

Una vez definida la forma de generar la geometría de terreno, corresponde texturizarlo. Existen muchas formas de texturizar un terreno tomando como base el heightmap que lo define, como por ejemplo texturizar según la altitud en un punto, la pendiente en un punto, incluyendo también técnicas de simulación, como por ejemplo la simulación de flujo de agua utilizando la gradiente en un punto. Un sitio de GitHub provee un buen conjunto de técnicas que pueden servir para texturizado en un paquete llamado **Terrain Topology** [25].

Paralelamente al texturizado de terreno, es necesario diferenciar las distintas zonas naturales que presenta nuestro país, con el fin de simular escenarios más cercanos a la realidad. Según la CORFO, Chile se define en cinco zonas naturales: Norte Grande, Norte Chico, Centro, Sur y Austral [26]. Cada una de estas zonas naturales posee características que las diferencian de las otras, por lo que es imperativo definir qué objetos poseen cada una de ellas para la etapa de generación y así contribuir a mejorar la experiencia.

Dado que el prototipo deseado corresponde a una simulación en base a zona natural, un tema indispensable a manejar es la adición de puntos de interés que sean reconocibles por los jugadores, con el objetivo de dar una sensación de familiaridad con el entorno y evitar que el escenario se torne monótono. La opción de implementación pensada va de la mano con el fin de crear una solución que se diferencie de las que existen en el mercado, por lo que, tomando en cuenta que se está utilizando el sistema de terrenos de Unity, es necesario crear un sistema propio.

Otro punto importante a mencionar es el valor que puede dar a la experiencia el hecho que el tiempo dentro del juego esté coordinado con el tiempo real. Es por esto que se sugiere la implementación de un **ciclo día-noche** que trate de imitar un ciclo natural para poder medir posteriormente el impacto en la experiencia de juego.

El objetivo final de la generación es poder suministrar en tiempo de ejecución los heightmap según la ubicación del usuario para así crear escenarios que sean una simulación basada en la zona natural del lugar donde se ubica el jugador, incluyendo un conjunto de puntos de interés que ayuden a dar más inmersión.

A continuación se presentan de forma más detallada los procesos de generación de los escenarios introducidos previamente.

4.1 Localización del jugador

El primer paso para identificar el escenario a generar consiste en conocer la ubicación del usuario que está utilizando el producto.

4.1.1 Ubicación

Luego de analizar distintas formas de obtener ya sea coordenadas o datos de ubicación, la mayoría de sitios respecto al tema sugiere el uso de consulta de ubicación por IP a través

de una consulta a alguna API. Existen varios sitios web que ofrecen el servicio de información de ubicación a través de IP mediante consultas a una API, de las cuales, de las que se investigó, se pueden listar las siguientes.

- ip api: http://ip-api.com/
- ipdata.co: https://ipdata.co/
- IP2Location: https://www.ip2location.com/

La mayoría de estos servicios son de tipo freemium, ya que ofrecen un acceso gratis para la utilización del servicio, pero de manera limitada en la forma de un máximo de consultas diarias o por minuto (dependiendo del sitio).

La característica común de estos servicios es que no aproximan la ubicación exacta del usuario, sino que rastrean la ubicación del proveedor de ISP con quien el usuario tenga contrato y, dependiendo de la forma en que se realice la aproximación, puede tener mucho margen de error, a tal punto que pueden variar mucho las ubicaciones devueltas.

En un principio, se escogió **ipdata.co** como la fuente de obtención de ubicación del usuario, pero en numerosas ocasiones se llegó a límites de cuota diaria durante fases de pruebas internas, por lo que se tuvo que cambiar por **ip api** [27]. La ventaja de ip api es que es de uso no comercial, no requiere API Keys para su integración, devuelve la información en varios formatos (incluyendo JSON) y su límite de consultas por minuto es lo suficientemente alto para no entorpecer las tareas de desarrollo y pruebas.

4.1.2 Obtención de Datos de Ubicación

Una vez seleccionado el servicio de localización, corresponde la integración con el sistema del prototipo en Unity.

Como se señaló anteriormente, el servicio de localización puede entregar la información en varios formatos, de los cuales se eligió **JSON**. Al visitar la página principal del ip api, nos muestra inmediatamente una respuesta de una consulta a nuestra IP con varios parámetros con este formato (ver figura 3).

```
{
    "query": "190.47.78.173",
    "status": "success",
    "continent": "South America",
    "continentCode": "SA",
    "country": "Chile",
    "countryCode": "CL",
    "region": "RM",
    "regionName": "Santiago Metropolitan",
    "city": "Nunoa",
    "district": "",
    "zip": "34033",
    "lat": -33.4667,
    "lon": -70.6,
    "timezone": "America/Santiago",
    "currency": "CLP",
    "isp": "Latin American and Caribbean IP address Regional Registry",
    "org": "VTR BANDA ANCHA S.A.",
    "as": "AS22047 Latin American and Caribbean IP address Regional
Registry",
    "asname": "VTR BANDA ANCHA S.A.",
    "mobile": false,
    "proxy": false
}
```


Figura 3. Consulta por defecto al ingresar al sitio de ip api.

De los parámetros listados, para uso del prototipo se recuperan los siguientes campos:

- query (IP)
- region (Abreviación de la región)
- regionName (Nombre de la región)
- city (Cuidad)

La recuperación de los datos se realiza mediante la ejecución de un **WebRequest** a la URL (API) **http://ip-api.com/json/** al iniciar la ejecución, luego procesar la respuesta los datos se almacenan en un objeto estático para futuro uso de ellos.

Procedimiento

1. Creación de **UnityWebRequest** con la URL de **ip api**.
2. Envío de la solicitud y espera por una respuesta del servidor
3. Al recibir la respuesta afirmativa, procesar respuesta JSON utilizando la librería **JSON.NET** [28].
4. Guardar los datos correspondientes en el objeto estático **PlayerData**.

Cabe mencionar que durante la etapa de desarrollo se vieron casos en que el ruteo correcto fue, por ejemplo, Concepción, Región del Biobío y con el correr del tiempo el ruteo era cambiado a Ñuñoa, Región Metropolitana de Santiago. Estas inconsistencias se acarrearon durante las siguientes etapas de desarrollo, por lo que se crearon medidas para no impactar demasiado la experiencia final.

4.2 Producción de Recursos de Generación de Terreno

Una vez definida la identificación de la ubicación del jugador, se necesita poder crear un terreno base según la información que proporciona el usuario por medio de la IP. Como se dijo anteriormente, se utilizarán los heightmaps para este fin, debido a que son una manera fácil para representar terrenos y, además, son parte importante del sistema de creación de escenarios, tanto en Unity, como en el desarrollo de videojuegos en general.

Es necesario recordar que Unity tiene ciertos requerimientos para trabajar con heightmaps, por lo que es necesario utilizar los software QGIS y GIMP para crear heightmaps con sentido para el proyecto.

Para el proyecto se define un perfil de heightmap para cada región del país, esto es:

- Que en el heightmap sea reconocible la forma de la región.
- Que el terreno producido sea de tipo isla (rodeado de agua).

Para lograr obtener heightmaps que cumplan estos objetivos, se utilizarán distintos datos dentro del entorno de **QGIS**, los cuales son los heightmaps obtenidos de la **SRTM** de la NASA y los datos vectoriales del sitio de la **IDE Chile** que corresponden a la división administrativa del país.

4.2.1 Extracción de datos geográficos

Los datos de la SRTM se obtuvieron específicamente del sitio del proyecto **SRTM Tile Grabber** [29] ya que ofrece una manera mucho más rápida y amigable de obtener tiles de los DEM que corresponden al territorio nacional. Estos datos ráster están muestreados a 90 metros (3 arc-segundo), donde cada tile tiene una resolución de 6000x6000 pixeles y tienen formato GeoTIFF. Existe una versión actualizada de esta herramienta de 30 metros (1 arc-segundo) [30][31], pero los tiles que conforman al territorio nacional son muchos en comparación a la versión de 90 metros com para ser trabajados con la misma facilidad en QGIS (ver figura 4).

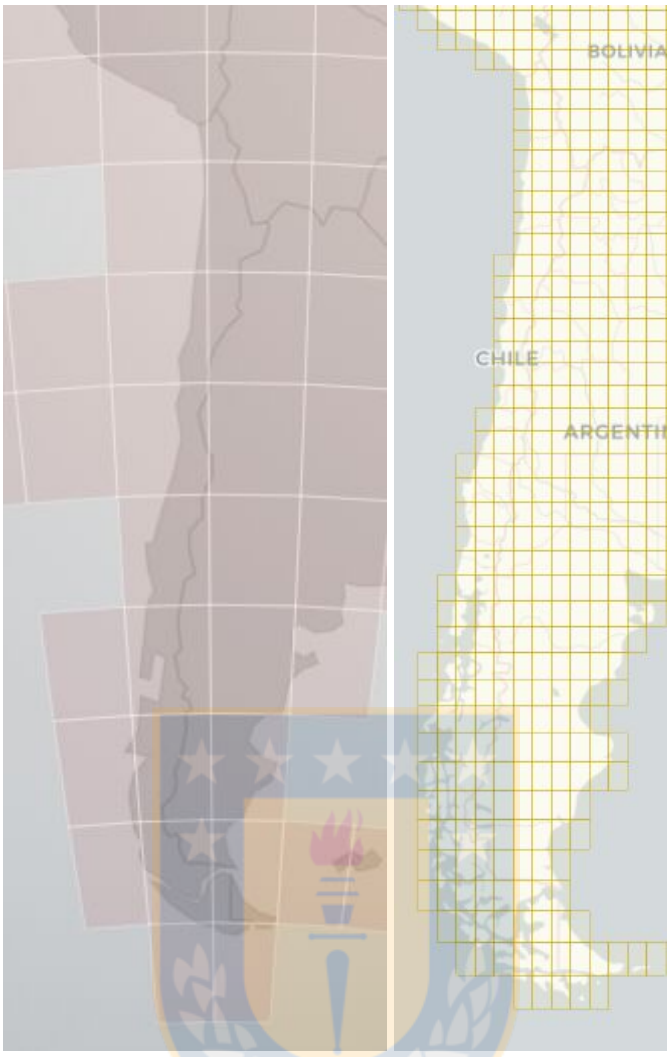

Figura 4. Tiles que conforman el territorio nacional, 90m con 17 tiles (izquierda) y 30m con 155 tiles aproximadamente (derecha).

Los datos vectoriales de la división administrativa del país se obtenidos del sitio web de la IDE Chile corresponden a un paquete con diferentes archivos .shp. Para obtenerlos basta solo con acceder al sitio web, ir a **Descargas > Capas > Límites > División Político Administrativa 2019**. El archivo descargado contiene capas con los límites administrativos para regiones, provincias y comunas, pero para este caso solo se utilizarán el archivo con los límites regionales.

Una vez obtenidos los datos, corresponde iniciar la manipulación en QGIS según el siguiente procedimiento.

Procedimiento

- Importar las capas ráster obtenidas desde SRTM Tile Grabber y la capa vectorial de la división administrativa de las regiones del país obtenidas desde la IDE Chile.
- Verificar que todas las capas estén en el mismo SRC. Para este caso se utiliza WGS84 (EPSG:4326) para todas.
- Por cada región:
    - Seleccionar las capas ráster que contengan a la región en cuestión y **Combinar** las capas para formar una sola imagen ráster.
    - Seleccionar (dejando marcada) la región (objeto espacial) de la capa vectorial.
    - Utilizar la herramienta **Cortar ráster por capa de máscara** para hacer un recorte de la capa ráster combinada con la forma de la región dada por la capa vectorial seleccionada.
    - Guardar Recorte resultante como imagen TIFF.

En la figura 5 se puede observar una superposición de la capa vectorial de la división administrativa regional con la capa ráster de los heightmaps.

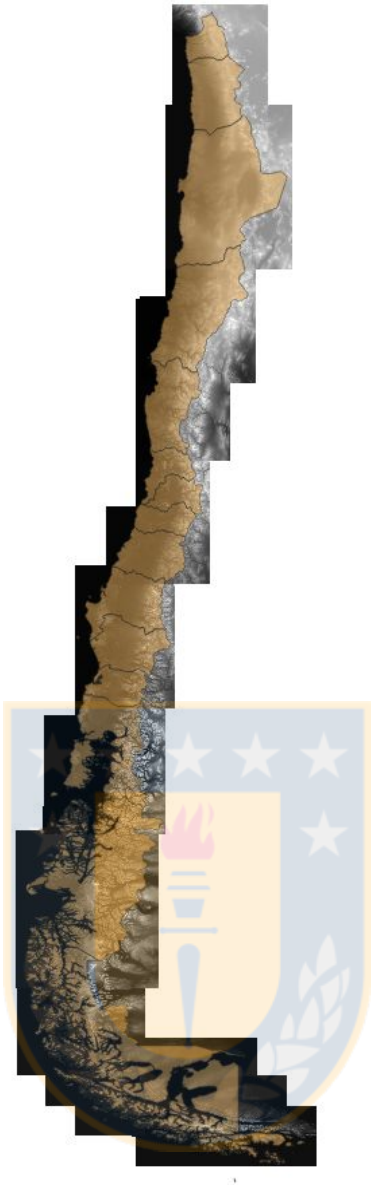

Figura 5. Capa división administrativa regional superpuesta sobre las capas ráster del territorio nacional en QGIS.

En varios casos ocurrieron errores en el recorte debido a un error de polígono inválido en la capa vectorial utilizada para el recorte. La solución a este problema fue aplicar una simplificación de geometría para esta capa, con un **tolerancia de 0.01**, lo suficientemente baja para mantener a simple vista la forma de la región involucrada. En la imágenes anteriores, se puede apreciar en la izquierda el polígono de corte en naranjo, sin simplificación, en verde el polígono resultante después de la simplificación y el resultado del corte con la capa simplificada (ver figura 6 para el resultado).

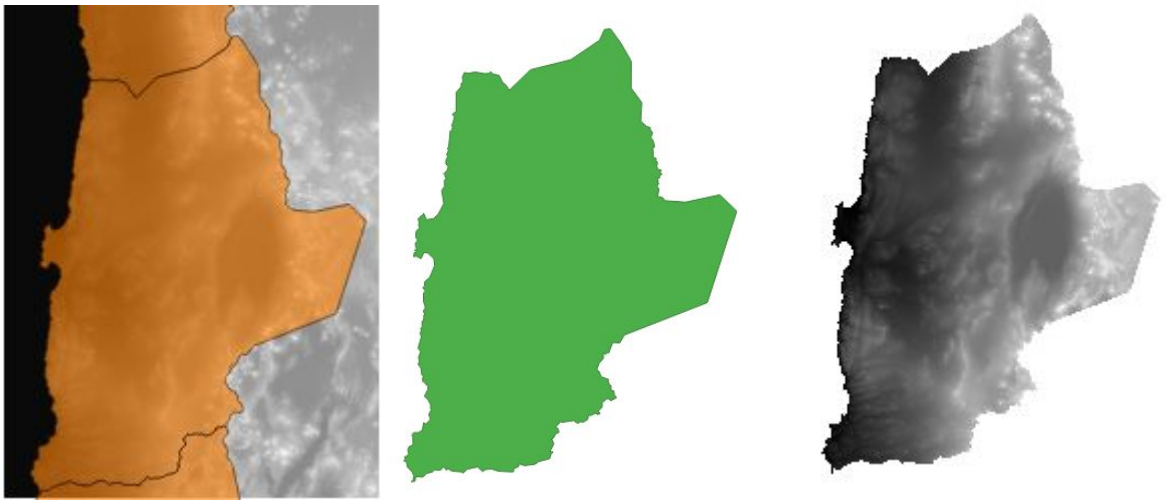

Figura 6. Superposición de capas Región de Antofagasta (izquierda). Simplificación de geometría de capa (centro). Recorte resultante (derecha).

Para el caso de las regiones del sur del país (Aysén y Magallanes), que poseen muchas islas, la solución al problema de polígono inválido no era aplicable debido a la complejidad del éste, por lo tanto fue necesario crear un polígono aproximado de forma manual que se ajustara a los límites de cada región, pero que mantuviese su forma característica.

4.2.2 Edición de Heightmaps

Como se dijo en un principio, para que las imágenes sean de utilidad, es necesario realizar una edición usando cualquier software de edición de imágenes como GIMP o Photoshop. Aquí, es **obligatorio** cumplir con las siguientes reglas para que el heightmap pueda ser procesado por Unity:

- Los heightmap deben tener el **mismo alto y ancho**.
- La dimensión del heightmap deber ser una **potencia de dos más uno** (513 x 513, por ejemplo) hasta 4097 x 4097.
- La imagen es exportada como archivo **.raw**.
- El archivo exportado desde GIMP debe ser en **escala de grises** con una **precisión de 8 o 16 bits**.

Para este proyecto se definen las siguientes reglas para la creación del escenario base (mesh que representa al terreno) a partir del heightmap editado:

- Resolución del heightmap de 2049x2049.

- El heightmap debe tener la región centrada. Los datos alrededor de la región son rellenados con valores de altura 0, que son representados por el color negro (**RGB(0,0,0)**). Esto le da la forma tipo isla al escenario (ver figura 7).
- Tamaño del terreno a generar en Unity: 8192 x 200 x 8192 (x,y,z).

Figura 7. Heightmap Final Región de Tarapacá.

Algunas complicaciones se presentaron durante la etapa de edición de heightmaps y las pruebas con la carga en tiempo real de los datos. Para más detalles, ver Anexo [10.2, 1].

4.3 Texturizado de Escenarios

El proceso de texturizado tiene la finalidad de producir una simulación de la vista aérea de cada región teniendo en cuenta las características de cada una de ellas en zona natural y geografía. Es por esto que está la necesidad de clasificar las regiones según estas características en grupos que permitan simplificar el proceso de texturizado, ya que del resultado de esta etapa depende la etapa siguiente de la adición de detalles y vegetación. Es por estas razones que se propone la clasificación de zonas naturales del país definidas por la CORFO.

4.3.1 Zonas Naturales

Chile posee mucha variedad de entornos, paisajes y climas que lo dotan de un rica diversidad de plantas y animales. Comenzando desde el norte, se tienen paisajes áridos, desérticos, semiáridos, mediterráneos, lluviosos, tundra, etc. a medida que se avanza hacia el sur. Para realizar una simulación más correcta es necesario tomar en cuenta esta variación de climas, pero para este caso el proyecto se centrará en una descripción más general de las zonas naturales como lo muestra la CORFO, que divide al país en cinco grandes zonas naturales:

- **Norte grande**: Clima desértico, acantilados costeros. Cordillera de la costa alta, depresión intermedia y cordillera de los Andes. Altiplano en los Andes, salares, depósitos de cobre y salitre en el interior.
- **Norte chico**: Clima semiárido, fusión de la cordillera de la Costa y la cordillera de los Andes, valles transversales de este a oeste en vez de depresión intermedia. Sin vulcanismo.
- **Centro**: Clima mediterráneo y mediterráneo continentalizado al interior, vegetación de matorral, cordillera de la Costa y cordillera de los Andes separadas hacia el sur. Depresión intermedia fértil.
- **Sur**: Clima templado oceánico con abundancia de lluvias y bosque valdiviano. Cordillera de la Costa y cordillera de los Andes de baja altura, depresión intermedia cerca del nivel del mar. Lagos glaciares, intensa actividad volcánica y geotermal.
- **Austral**: Clima templado-lluvioso, frío y bosque subpolar magallánico. Paisaje glacial; cordillera de la Costa se compone de islas, depresión intermedia bajo el nivel del mar. Fiordos y campos de hielo.

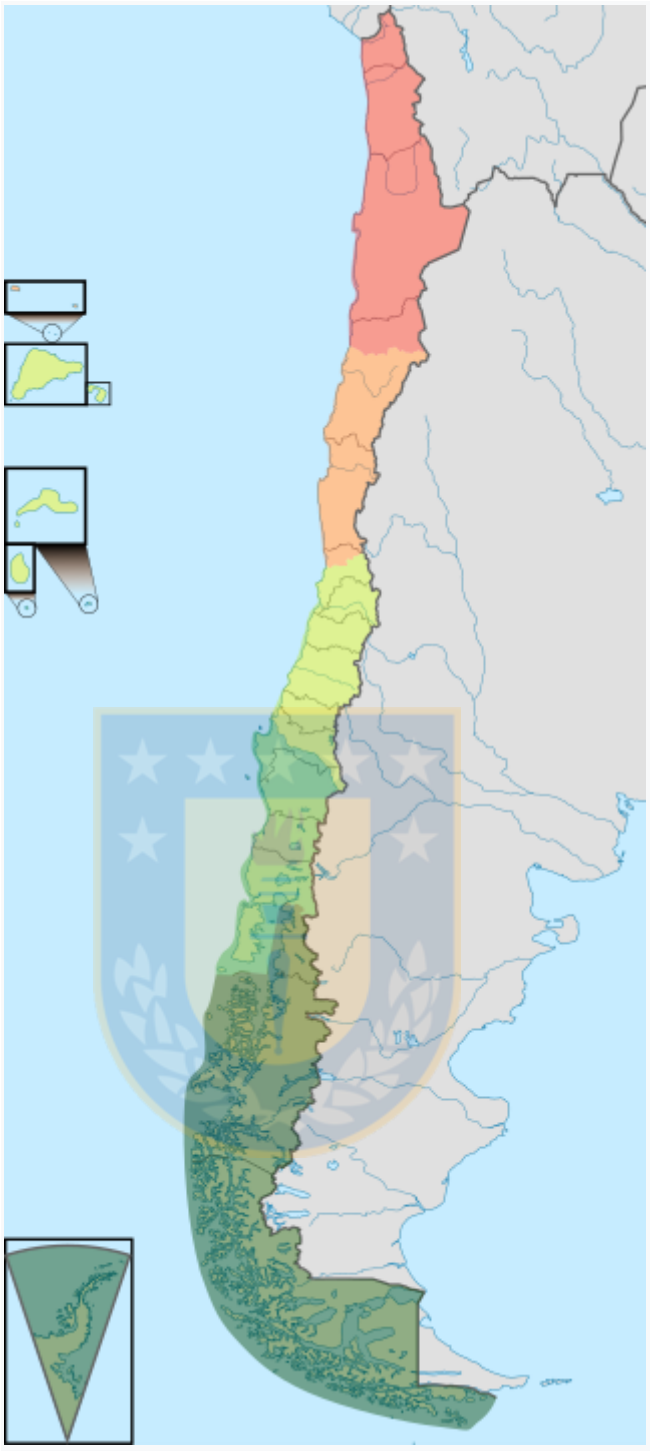

Figura 8. División de zonas naturales (regiones naturales) según la CORFO [26]

A partir de la figura anterior es posible apreciar que hay regiones que pertenecen a dos zonas naturales (ver figura 8), pero para este proyecto se hará una simplificación, dejando a las regiones que presenten dicha situación clasificadas como parte de la zona natural con mayor presencia. Así, las zonas naturales quedan compuestas por las siguientes regiones:

- **Norte Grande**
    - XV Región de Arica y Parinacota.
    - I Región de Iquique.
    - II Región de Antofagasta.
- **Norte Chico**
    - III Región de Atacama.
    - IV Región de Coquimbo.
- **Centro**
    - V Región de Valparaíso.
    - RM Región Metropolitana de Santiago.
    - VI Región del Libertador General Bernardo O'higgins.
    - VII Región del Maule.
    - XVI Región del Ñuble.
    - VII Región del Biobío.
- **Sur**
    - IX Región de la Araucanía.
    - XIV Los Ríos.
    - X Los Lagos.
- **Austral**
    - XII Aysén del General Carlos Ibáñez del Campo.
    - XIII Magallanes y la Antártica Chilena.

4.3.2 Estructura de una zona natural

Para poder implementar zonas naturales según la región, es necesario armar una estructura de datos para almacenar todos los objetos relacionados a ella y que, además, se ajusten al sistema de terreno de Unity. Para ello, se considera crear un sistema de capas para cada zona, donde cada capa incluye lo siguiente:

- Textura de capa.
- Árboles de capa.
- Hierba de capa.
- Objetos de detalle de capa.

Además, dadas las características de cada zona natural, se tienen variaciones en la altura de las cadenas montañosas, por lo que es necesario añadir opciones de altura para ellas, con fines de texturizado principalmente.

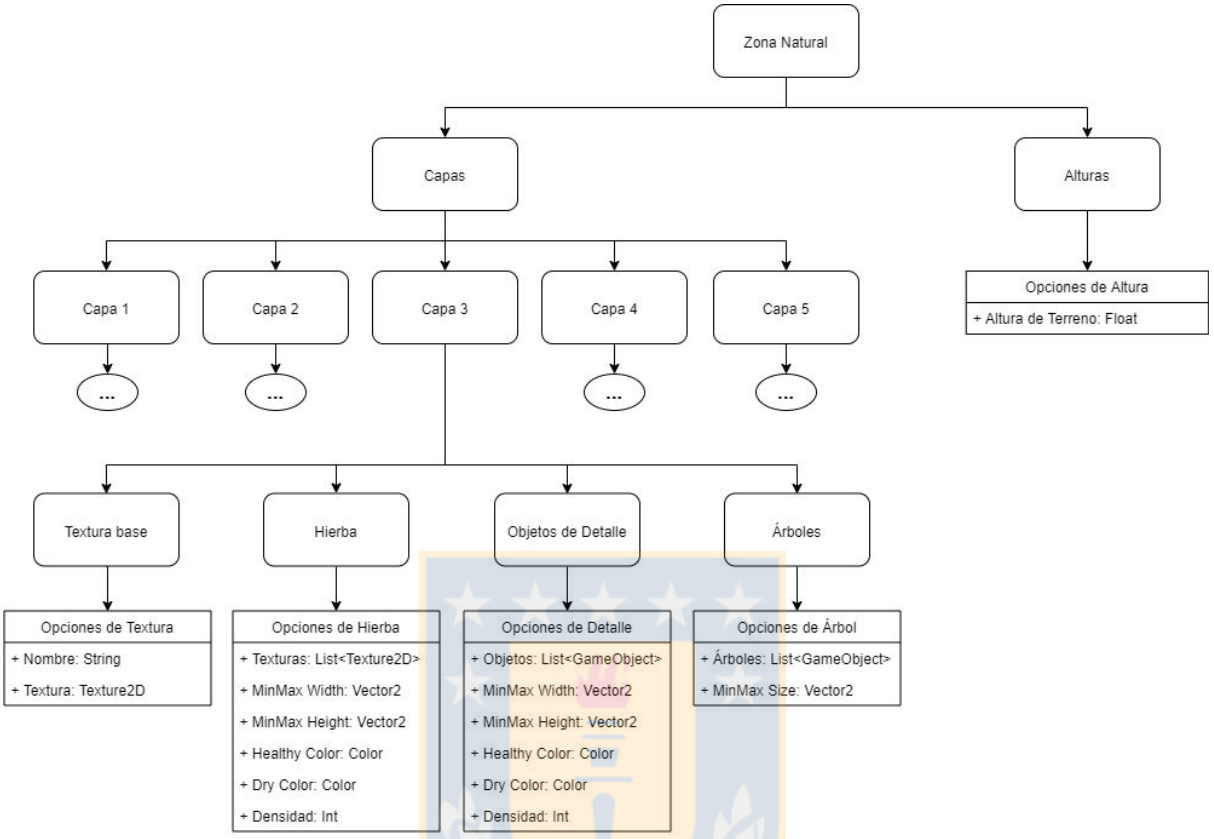

Figura 9. Diagrama de Estructura de Zonas Naturales.

Con esta estructura (ver figura 9), se crean perfiles que permiten almacenar toda esta información de forma permanente, para no tener que volver a hacer nuevos objetos en el caso de querer reutilizar el código.

4.3.3 Proceso de Texturización de Escenarios

Ahora, para la parte técnica, se tiene que tener en cuenta la capacidades del Terrain Engine de Unity. Particularmente, hablando del objeto TerrainData, las texturas que se utilizan en el terreno se almacenan como capas de texturas. Esta estructura se denomina **Alphamap** (Weightmap) y corresponde un array tridimensional en donde las primeras dos dimensiones se refieren a las coordenadas de un valor de la matriz en el Alphamap, el tercero corresponde al número de la capa y el valor almacenado corresponde al nivel de influencia de la textura en ese punto con su valor dentro de un rango entre 0 y 1.

La aplicación de texturas al terreno está fuertemente basado en altura e inclinación, que vienen dadas por el heightmap utilizado en la etapa de generación de la geometría de terreno. La idea es dar un porcentaje de influencia según las variables indicadas tratando de imitar cómo luciría un terreno natural.

Reglas de Texturizado

Para llevar la idea anterior a una implementación en Unity es necesario definir la influencia de cada textura en cada caso. Para ello, se utilizan operadores condicionales que asignan un valor alto dependiendo de si se cumple la condición establecida, en caso contrario se asigna un valor bajo. Con esto se asegura que siempre sea solo una textura la que tenga una mayor influencia en cada punto.

Ejemplo de reglas de texturizado:

- La capa 1 tiene influencia constante.
- La capa 2 es más notoria a baja altitud.
- La capa 3 es más notoria en terreno plano con altura mayor a cero.
- La capa 4 aumenta con la altura y pendiente.
- La capa 5 representa al terreno plano fuera (o dentro) de la forma de la región, sólo cuando la altura es igual a cero.

Cada una de las reglas de ejemplo anteriores posee un operador condicional para asignar su valor de influencia en cada caso.

Uso de Flow Maps

Si observamos zonas naturales, podemos observar que hay una cantidad notoria de área verde por donde pasan ríos u otros flujos de agua como la lluvia. La influencia de estos flujos hacen que una montaña, por ejemplo, pase de ser un páramo más o menos estéril, a un contraste entre lo seco y lo vivo (ver figura 10).

Figura 10. Vista satelital de El Yugo, Región Metropolitana de Santiago (Google Maps). En la imagen se puede apreciar como hay una mayor concentración de áreas verdes mientras se sigue la dirección del flujo.

Si consideramos los distintos tipos de simulaciones posibles de realizar con los computadores, nos encontramos con el caso de la simulación de erosión hídrica, que en el ámbito de la creación de escenarios es ampliamente utilizada para dar un moldeado y suavizado natural a las formaciones terrestres bajo la influencia de agua, así como para crear ríos siguiendo la dirección del flujo. Sin embargo, se puede utilizar para apoyar el texturizado de los escenarios, tratando simular áreas más verdes en ambientes mayoritariamente secos con ayuda de los **Flowmaps**.

Con la finalidad de enriquecer el resultado del proceso de texturizado, se incorporan estos recursos dentro del conjunto datos utilizados para aportar a la texturización. La simulación utilizada para crear estos mapas de flujo es dada por el paquete **Terrain Topology** [25] y está basada en un descenso de gradiente sobre el heightmap de terreno.

Se realizaron algunas modificaciones al paquete para poder utilizar los heightmaps de las regiones con las cuales se está trabajando (ver figura 10). El objetivo es generar una imagen de escala de grises por cada región para luego poder utilizarla en el proceso de texturizado.

La imagen es almacenada como archivo .png y, al igual que los heightmap, es una imagen en escala de grises (ver figura 12) y es posible acceder al valor de cada pixel a través de la clase Color, luego de ser procesada por la clase **Texture2D**, que, utilizando la función GetPixels, retorna un array con todos los píxeles de la imagen.

Figura 11. Heightmap Región de Tarapacá.

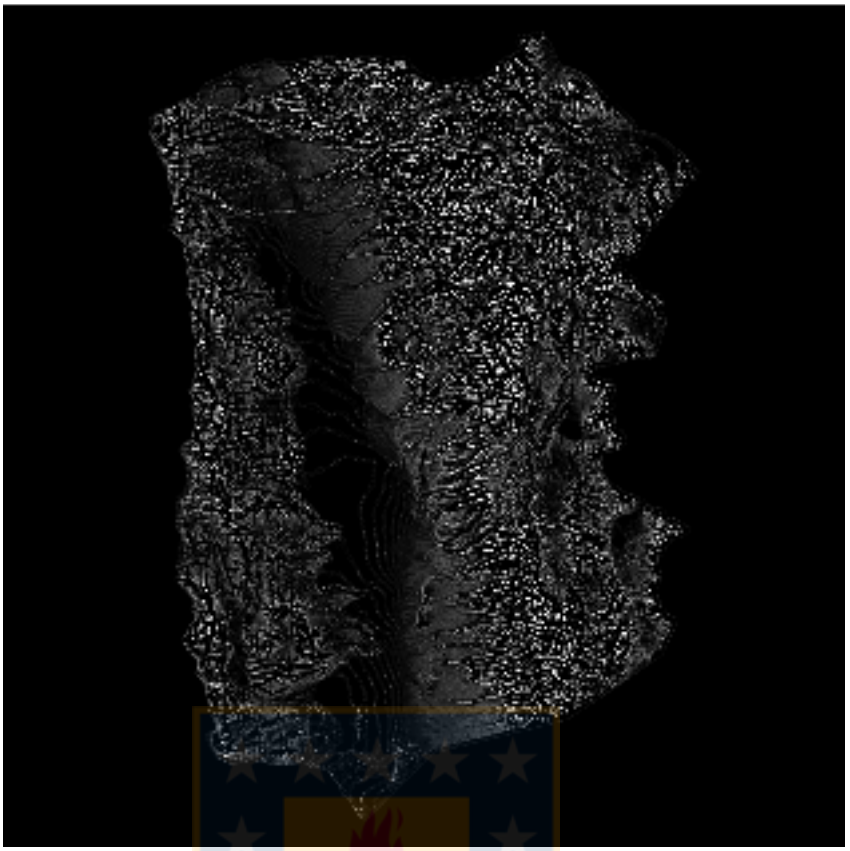

Figura 12. Flow Map Región de Tarapacá obtenido a partir del Heightmap.

La documentación de Unity indica cómo realizar un blending de texturas a través de C# de forma básica [32] y el blog de Alastair Aitchison da un ejemplo de como hacer un texturizado a través de scripts utilizando operadores condicionales, considerando más capas de textura [33].

El procedimiento general se define de la siguiente manera:

1. Crear un array tridimensional para almacenar el nuevo Splatmap con dimensiones 2048x2048x6, es decir alto x ancho x número de capas de textura. Como aclaración, el siguiente bucle que recorre este array tridimensional lo realiza bajo el concepto de loop unrolling, ya que se trabaja con todas capas al mismo tiempo para asignar el valor de influencia.

2. Para cada iteración

- 2.1. Asignar el valor de influencia de capa para las seis capas.
- 2.2. Sumar los valores de influencia para producir un factor.
- 2.3. Para cada capa:
  - 2.3.1. Normalizar el valor de influencia obtenido dividiendo el valor de influencia por el factor obtenido anteriormente.

2.3.2. Guardar el valor normalizado en la coordenada correspondiente del Splatmap de la capa.

3. Aplicar el nuevo Splatmap al terreno.

4.3.4 Resultado del Texturizado

Al aplicar los métodos de texturización descritos anteriormente, se puede apreciar el siguiente resultado (figura 13, comparativa con la figura 14).

Figura 13. Vista superior texturizado de la XIV Región de Los Ríos.

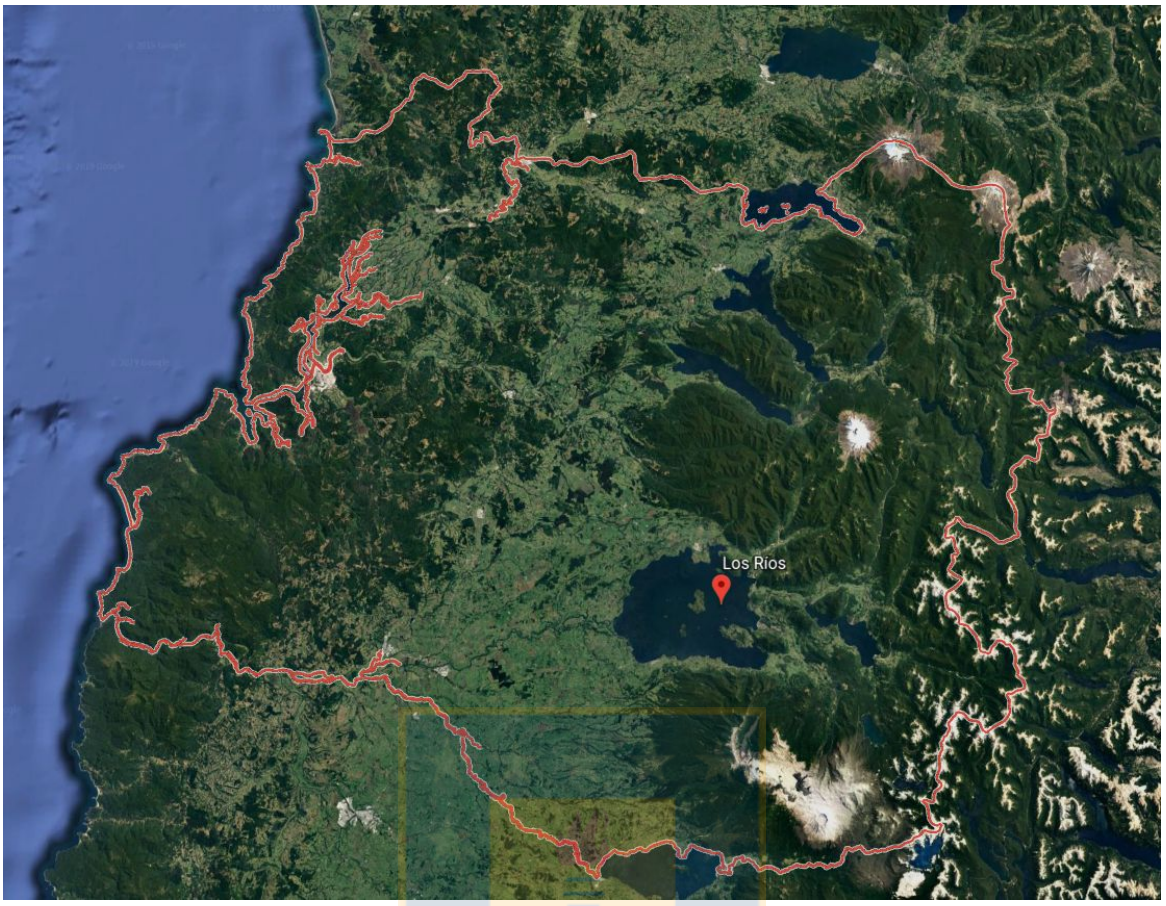

Figura 14. Vista superior de la XIV Región de Los Ríos (Google Earth).

4.4 Distribución de Elementos Decorativos en el Terreno

En Unity, los detalles de terreno corresponden a ciertos elementos decorativos que lo convierten en un entorno más inmersivo para el jugador, como los árboles, arbustos, rocas, hierba y pequeñas plantas plantas similares. Estos elemento se clasifican en las siguientes categorías:

- **Trees** (Árboles): abarcan árboles y arbustos, pero básicamente cualquier objeto producido por el editor de árboles de Unity o la herramienta externa SpeedTree.
- **Details** (Detalles): corresponde al pasto y objetos pequeños como rocas, ramas, etc. El hierba es un tipo general para describir distintos tipos de plantas que puedan ser similares a la hierba (esto incluye flores) y corresponden a texturas, mientras que los objetos de detalle corresponden a GameObjects (prefabs).

Estos tipos de objeto tienen parámetros para variar la forma en que son distribuidos por el escenario, como rango de altura, rotación, variación de color, densidad, etc.

La idea es ubicar los elementos decorativos considerando la zona natural de cada región y su distribución de texturas en el **Splatmap** obtenido en la etapa de texturizado. Para esto, se asigna uno o más tipos de árbol y uno o más tipos de detalles a cada capa de textura, según lo indicado en el perfil de zona natural correspondiente.

La distribución de objetos decorativos es realizada de una forma relativamente aleatoria en base al heightmap base del terreno y el Alphamap para ver la influencia de las Texturas y se resume en dos procesos importantes: distribución de **Trees** y distribución de **Details**.

Los Trees definen su distribución con respecto al **Alphamap** (ya que depende de la textura), que se representa como una matriz de 2048x2048, mientras que los Details definen su distribución y densidad por un **Detail Map**, que Unity la representa para este caso como una matriz de enteros de dimensión 4096x4096, donde cada valor de la matriz almacena un valor de entre 0 y 16, lo que representa la densidad del detalle en un área. Sin embargo, ambos procesos hacen uso del **Heightmap** y **Alphamap**, ya que su distribución debe tomar en cuenta la forma del terreno y la textura con mayor presencia en cada punto.

Ambos procesos disponen de un **diccionario** que contiene un id de la capa como **key** (número de la capa) y un objeto llamado **LayerMap**, que contiene las IDs de los objetos decorativos, como **value**. Cada Layer Map contiene un array para guardar IDs de árboles, uno para texturas de pasto y otro para objetos mesh decorativos y su presencia es necesaria ya que el sistema de terrenos de Unity almacena todos los árboles en un mismo array tipo **TreePrototype** [36] y en otro array de tipo **DetailPrototype** [37] almacena todas las texturas de pasto y mesh de detalle, por lo que no se puede saber de antemano cuales Trees, Details corresponde a cada capa.

4.4.1 Procedimiento de distribución de Trees

Para la distribución de Trees se sigue el siguiente procedimiento.

1. Por cada capa en la estructura de zona natural
- 1.1. Por cada elemento del Alphamap
  - 1.1.1. Verificar si la capa tiene Trees asignados para distribuir en el terreno. En caso contrario, seguir con el próximo elemento.
  - 1.1.2. Si la cantidad de Trees asignados es mayor a 0, calcular una probabilidad de aparición de un Tree (1/36 de probabilidad, es decir

2.7% de 2048x2048 lugares posibles, esto se hace para evitar una sobrepoblación de Trees que afecten mucho el rendimiento del juego). En caso contrario, seguir con el próximo elemento.

1.1.3. Si se determina que se debe colocar un Tree, verificar que el valor de influencia de la textura de la capa es el mayor. En caso contrario, seguir con el próximo elemento.

1.1.4. Verificar si el terreno en ese punto no tiene una pendiente muy pronunciada (pendiente menor a **0.9**). En caso contrario, seguir con el próximo elemento.

1.1.5. Determinar la cantidad de Trees a ubicar en un radio del punto, con el fin de producir bosques (definido por el parámetro Tree Density en el menú principal del prototipo).

1.1.6. Por cada Tree a ubicar

1.1.6.1. Seleccionar un prefab de forma aleatoria de los disponibles en el Layer Map.

1.1.6.2. Instanciar el Tree en el escenario.

2. Actualizar el terreno para aplicar los cambios.

4.4.2 Procedimiento de distribución de objetos de tipo Detail.

En el caso de la **Distribución de objetos tipo Detail** se sigue es siguiente procedimiento.

1. Por cada capa en la estructura de zona natural

1.1. Verificar si la capa tiene objetos tipo Detail asignados para distribuir en el terreno. En caso contrario, seguir con la siguiente capa.

1.2. Crear una matriz de enteros para generar un **Detail Layer** para la capa.

1.2.1. Por cada elemento del **Detail Layer** creado

1.2.1.1. Si el valor es mayor a 0 y la inclinación menor a 0.9 asignar el valor de densidad de detalle definido en la capa de zona natural. En caso contrario asignar valor de densidad 0.

1.3. Guardar detail map generado.

2. Por cada Detail presente en el Layer Map de la capa

2.1. Asignar el detail layer correspondiente al objeto detail según la indicación del Layer Map.

2.2. Guardar el detail layer en el Detail Map.

3. Actualizar el terreno para aplicar los cambios.

4.5 Manejo de Puntos de Interés

Una vez generado el escenario, es necesario poblarlo con puntos de interés de cada región. Para ello se utiliza la API Overpass, realizando web request al servidor con los parámetros necesarios.

Los parámetros de consulta serán realizados en base a **tags** de nodos y los distintos valores que pueda tomar según los disponibles en la documentación de OSM [39].

Hay que tener en cuenta que, debido a la naturaleza del prototipo, no todos los nodos disponibles son de utilidad. Por ejemplo, el valor **hotel** del tag tourism no es de utilidad ya que el prototipo se enfoca en lugares poblados, naturaleza, geografía no de un nivel de granularidad bajo y común como una calle, edificios, estatuas, etc.

4.5.1 Obtención de datos

Debido a las diferencias entre zonas naturales, regiones y otros factores, los sets de tags recuperados serán definidos por zona natural pues la densidad de nodos entregadas para distintas regiones puede variar mucho dependiendo de la zona en la que se encuentren. Por ejemplo, al recuperar las ciudades y pueblos en la Región del Biobío, se retornan 40 elementos, mientras que la misma consulta para la región de Tarapacá devuelve 5 elementos, lo que deja el escenario con muy pocos nodos.

La selección de puntos de interés se define para los siguientes tags o tipos:

- Tag **place:**
    - Ciudades.
    - Pueblos.
    - Poblados.
- Tag **water**
    - Lagos.
    - Lagunas.
    - Embalses de tamaño considerable.
- Tag **tourism**
    - Playas
    - Atracciones turísticas relacionadas a la geografía y naturaleza como miradores y campings.

- Tag **peak**
    - Cerros.
    - Montañas.
- Entre otros.

La selección de puntos de interés está hecha en base a los tags descritos en la sección Map Features de Open Street Map Wiki [39], escogiendo los tags que están más relacionados a la naturaleza y que posean un nombre, ya que existen muchos nodos, por ejemplo montañas y cerros, que aparecen sin un nombre en Overpass, por lo que no son de utilidad en el prototipo.

Formato de Consulta

Las consultas de nodos a Overpass tienen el siguiente formato:

```
node[place="city"](area.a);
node[place="town"](area.a);
...
```

En la sección de Anexos [10.2, 2] se describe una problemática con el tiempo total de consulta a Overpass.

4.5.2 Ubicación de POIs en el Terreno

Cada elemento de estos tipos de punto de interés tiene un objeto Prefab asociado para instanciar en la escena. Ubicar los prefab de cada punto de interés en una ubicación en el terreno lo más parecida a la original supone un gran desafío, cuyas complicaciones son:

1. La carencia de algoritmos, métodos, frameworks que integran datos geográficos con el sistema de terreno de Unity.
2. Los distintos niveles de zoom realizados a los DEM durante el proceso de recorte en QGIS debido a las dimensiones de cada región.
3. Las distintas proporciones de escalado de imagen durante la etapa de edición de los DEM en GIMP (también debido a las dimensiones de cada región).

Procedimiento

1. Web request de los puntos de interés.
    1.1. Creación de webrequest con URI de Overpass Turbo más la consulta en Overpass QL y envío de petición por medio de UnityWebRequest.
2. Procesamiento de la respuesta.
    2.1. Respuesta desde el servidor en formato JSON.
    2.2. Guardar resultados en listas de un objeto estático para hacer uso de ellas en la generación del escenario.
    2.3. Eliminar nodos muy alejados del centro de la región. Esto se realiza para descartar nodos que se encuentren en islas muy alejadas de la región, ya que no se abordan en este trabajo, como por ejemplo la Isla de Pascua.
3. Mapeo LL a XYZ de las coordenadas de los nodos (Mapbox API).
    3.1. Utilizando la API de Mapbox se pueden realizar traducciones de Latitud - Longitud al Sistema cartesiano XYZ de Unity para cada nodo.
    3.2. Ajustar traducción al tamaño del terreno (8192x8192).
4. Fijación de puntos de interés a la superficie del terreno por medio de Raycast.
    4.1. Los nodos aparecen en el escenario a una altura mayor a la del terreno, distribuidas en un plano.
    4.2. Por cada punto, utilizando la clase Raycast de Unity, se lanza un rayo desde la posición del nodo hacia abajo hasta que el rayo se encuentre con el collider del terreno.
    4.3. Se almacena el punto de colisión.
5. Ubicación de prefabs en el punto.
    5.1. En cada punto de colisión obtenido se ubica un prefab.
    5.2. El prefab varía según el tipo de nodo.

En el caso de los POI que representan ciudades, pueblos y poblados, cada prefab tiene un instanciador de construcciones (cabaña, casa, edificio) con cierta probabilidad de aparecer, donde los tipos de modelo y la cantidad de instancias alrededor del marcador varía según el tipo de POI, es decir:

- Para Ciudades: hasta **24** construcciones, casas, cabañas, y apartamentos disponibles.
- Para Pueblos: hasta **8** construcciones, casas y cabañas.

- Para Poblados: hasta **4** construcciones: casas, cabañas.

Para poder realizar la colocación de los puntos de interés devueltos por Overpass, se tuvo que lidiar con varias complicaciones que requieren ajustes manuales que toman bastante tiempo. El detalle de los procedimientos realizados se puede ver en la sección de Anexos [10.2, 3].

4.5.3 Ubicación de Cuerpos de Agua en el Terreno

El caso de los cuerpos de agua es más complejo, ya que no están formados solo por un nodo (punto), sino que por un conjunto de ellos. Dependiendo el caso, un cuerpo de agua puede ser representado como una vía o una relación. Una vía es un conjunto de nodos que generan un polígono que representa la forma de un objeto y una relación es un conjunto de vías y/o nodos que forman un polígono mayor con más características.

Método desarrollado

Dentro de los datos que devuelve la consulta detallada en el anexo [10.2, 4], se puede obtener la bounding box que contiene a la figura del lago. La bounding box contiene cuatro valores que indican la latitud-longitud mínima y máxima dentro de la cual los puntos del polígono debieran estar.

De la bounding box se obtiene el punto medio que es equivalente a obtener el centro del polígono en el método 1 [10.2, 4] y se busca por un punto que tenga pendiente cercana a 0 y su altura sea mínima dentro de la bounding box en la dirección dada por un vector desde el punto medio hacia el punto de máxima latitud-longitud de la bounding box. Al obtener el punto deseado, se ubica el mesh en el punto (ver figura 15 para una visualización del método descrito a continuación).

Procedimiento

1. De la bounding box obtenida, calcular el punto medio y seleccionarlo como punto de partida.
2. Utilizando Raycast, instanciar el prefab de agua en el punto medio.
3. Obtener la submatriz del heightmap contenida dentro de la bounding box.
4. Por cada valor de la submatriz de alturas, obtener el gradiente del terreno en el punto.

5. Si el valor de la gradiente en el punto es suficientemente bajo (terreno plano, gradiente menor a 0.1), verificar que:
    a. El punto no esté a una altura mayor que el punto de partida. Esta restricción evita que se consideren puntos que sean planos sobre la posición inicial.
    b. El punto no supere una distancia máxima al punto de partida. Esta restricción evita considerar los puntos planos del terreno que están fuera de los límites de la región
6. Almacenar en una lista la posición en el espacio (Vector3) de los puntos que cumplas ambas condiciones.
7. Obtenido el conjunto de puntos que cumplen con la condición, calcular el promedio y la desviación estándar de las alturas de los puntos de la lista y remover los puntos que están fuera del rango admisible de la desviación. Con esto, se descartan los puntos en terreno plano que están muy abajo o muy arriba de la altura de los datos útiles.
8. Obtener punto con máxima altura dentro de la lista restante y realizar un Raycast desde una altura prudente directamente hacia abajo. Con esto se asegura que el prefab quede ubicado por sobre la superficie del terreno.
9. Devolver la altura resultante más un desfase positivo.

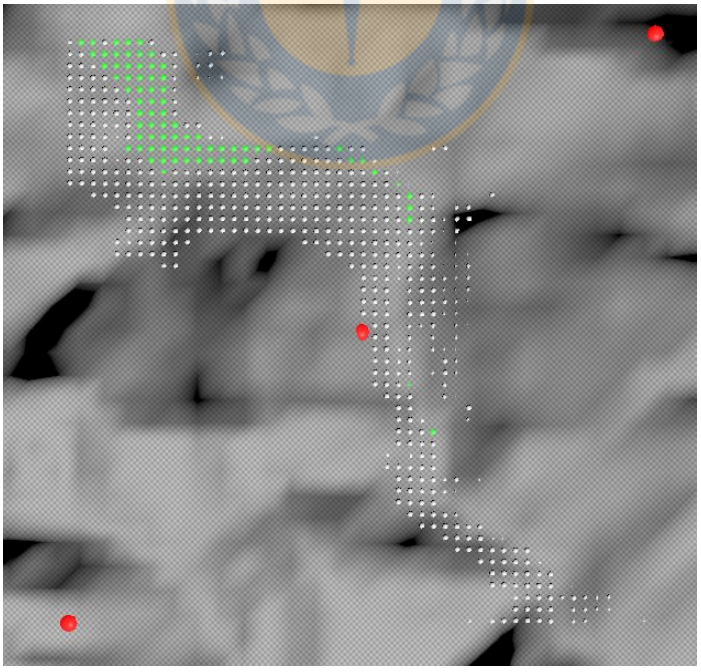

Figura 15. Visualización del algoritmo de corrección de altura. Puntos rojos: límites de la bounding box y punto medio, puntos blancos: puntos del heightmap de menor altura que el punto medio, puntos verdes: puntos en terreno plano que cumplen las condiciones.

La solución final se define como una combinación de tres métodos (ver anexo para más detalles del método 1 [10.2, 4 Método 1] y método 2 [10.2, 4 Método 2]), ya que comparten problemas comunes, por lo tanto, el procedimiento se reduce a utilizar el procedimiento del tercer método ya explicado, crear el mesh de cada masa de agua a partir de los puntos obtenidos de Overpass (ver anexo para ver el procedimiento en detalle [10.2, 4 Método 2]) y buscar la altura a la cual el mesh debe ir posicionado cuando el punto medio de la figura está fuera de ella. Aún así, no todos los posicionamientos son exactos, ya que también depende que tanto la escala como los offset definidos por región sean precisos.

4.6 Ciclo Día-Noche Coordinado

Con el fin de agregar más inmersión al entorno desarrollado, se propone la implementación de un sistema de ciclos dia noche que sea guiado en tiempo real, es decir, que esté coordinado con la hora actual del equipo del usuario. Cabe mencionar que en este sistema no se tiene considerada la evolución en la duración de los día según la estación del año.
Para llevar a cabo la implementación del sistema base, se eligió un asset de la tienda de Unity llamado **Simple Day and Night Cycle** [41]. Este sistema permite modificaciones a los scripts originales para adaptarlo a las necesidades de cada proyecto. El paquete descargado utiliza perfiles para cada fase del día, noche y uno solo que representa amanecer / atardecer, donde cada perfil define un color para el cielo, uno para el horizonte y otra para el ecuador predefinidos que se pueden modificar pero no otorgan un buen resultado en las transiciones de fase.

Para lograr el ciclo en tiempo real, cercano a un ciclo día-noche real se definieron las siguientes modificaciones.

1. Adecuar la el ciclo del día a la hora actual del sistema del usuario.
   Por defecto, el sistema tiene un control interno que representa un día, donde cada día posee sesenta segundos. Este valor puede ser modificado, entonces, por una simple fórmula matemática, se obtiene que 1 día equivale 86.400 segundos. Este valor en segundo es convertido a un intervalo entre 0 y 1 para que sea utilizado por el sistema.
2. Corrección de fases dia, noche, atardecer / amanecer que presentan transiciones abruptas de color.
   Para corregir los cambios abruptos de color entre fases fue necesario desechar los perfiles predefinidos por algo que permitiese una mezcla suave de colores entre transiciones de fase, es decir una gradiente de color. Como explica el desarrollador

del canal One Wheel Studio en su video Sky Box Module [42], se puede utilizar una gradiente de color (incluida en Unity Engine) para definir los colores en cada momento del día, como ya se hacía en el paquete por defecto, mezclado con las Procedural Skybox de Unity para dar un mejor efecto de transición de fases (figura 16). Como las gradientes están definidas en un rango 0% a 100% e internamente se representa en un rango entre 0 y 1, se puede evaluar el valor del tiempo actual en segundos mapeado al rango [0,1] en la gradiente y obtenemos un color para ese tiempo.

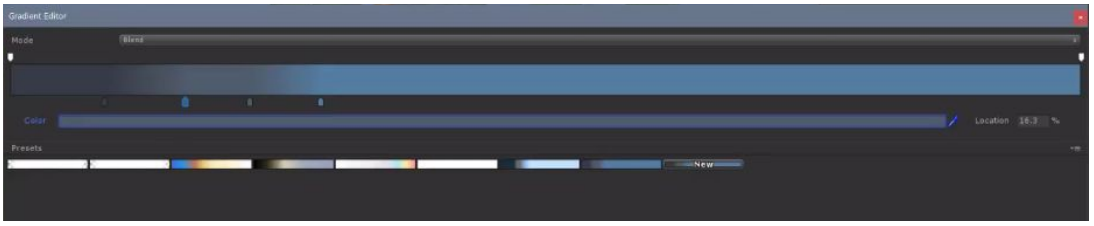

Figura 16. Editor de Gradiente en Unity.

3. Corrección de cambios de color abruptos para la luz direccional del escenario (que simula el sol y la luna para cada fase).

De manera análoga a la corrección 2, se utilizó una gradiente para definir la variación de color de la luz direccional durante el ciclo. Según el video del canal [42], para lograr un mejor efecto visual se debe hacer lo siguiente (ver figura 17):

- Durante el día, el color de la luz direccional del escenario debe tener un tono ligeramente amarillo.
- Durante la noche, el color de la luz direccional del escenario debe tener un tono ligeramente azul.
- Durante el amanecer (atardecer), el color de la luz direccional del escenario debe tener un tono anaranjado.

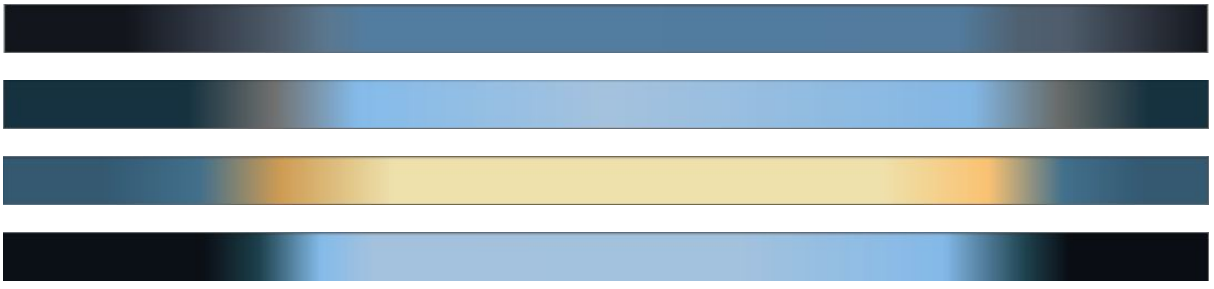

Figura 17. Gradientes utilizados en el ciclo día noche. Desde arriba hacia abajo: gradiente de cielo, gradiente de horizonte, gradiente de tono de luz direccional (sol y luna) y gradiente de color de niebla atmosférica.

4. Corrección de cambios de intensidad de la luz direccional en cada fase.

   Para otorgar una intensidad de luz correcta en cada fase y momento del ciclo es importante ajustar los niveles de la intensidad de la luz direccional que utiliza Unity y la Procedural Skybox. En el mundo real, la intensidad de la luz varía con transiciones relativamente suaves entre cada fase, al contrario de los valores por defecto del paquete Simple Day and Night Cycle. En un artículo no muy relacionado al objetivo de este proyecto, se trata el cómo los cambios de intensidad de la luz gatillan ciertos comportamientos en los animales [44]. En este artículo, se muestran las variaciones de intensidad de la luz a lo largo de las 24 horas del día para las estaciones de invierno y verano, especificando los rangos que comprende cada fase en cada gráfico. Esta información es sumamente útil para simular la intensidad de la luz durante una partida en el prototipo, por lo que se utilizó una curva para replicar la forma del gráfico **b** (ver figura 18), pero de una forma más simple, ya que en el proyecto no se considera la variación entre la duración del día y de la noche a medida que avanzan las estaciones.

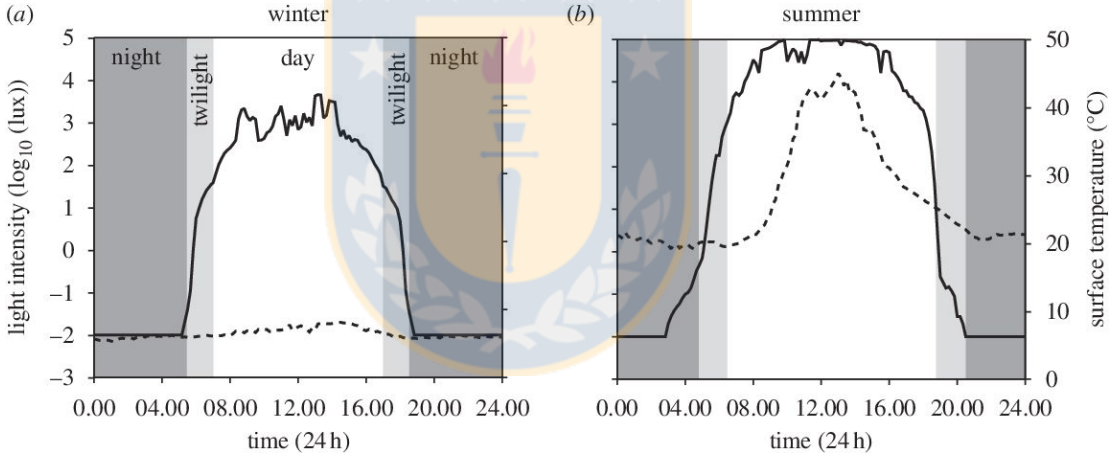

Figura 18. Variación de la intensidad de la luz (línea sólida) en invierno (a) y verano (b) [44].

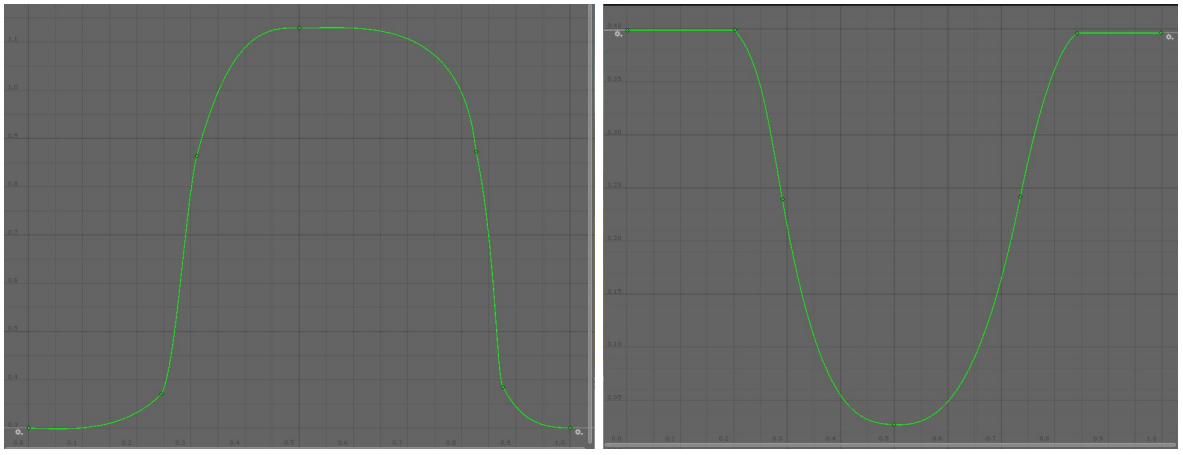

Figura 19. Curvas de intensidad de luz direccional para el sol (izquierda) y la luna (derecha).

Se presentaron inconvenientes con la puesta de sol / salida de la luna que se detallan en el Anexo [10.2, 5].

4.7 Resultados de Generación de Escenarios

En esta sección se presentan imágenes de los escenarios generados con el sistema implementado, incluyendo la diferenciación de horas del día (figuras 20 a 29). Además se incluyen algunas imágenes de la UI del prototipo, mostrando las opciones de generación y selección de región del menú principal (figura 30), sistema de construcción (figura 31) y sistema de cambio de hora (figura 32).

4.7.1 Norte Grande

Figura 20. Ocho de la mañana en mapa generado, Norte Grande.

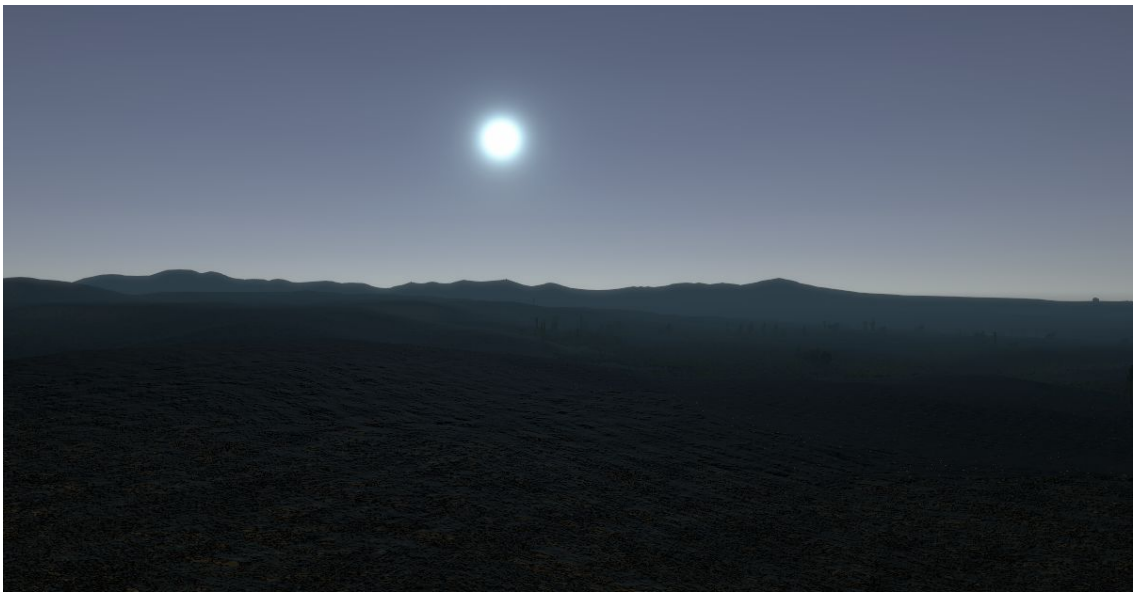

Figura 21. Nueve de la noche en mapa generado, Norte Grande.

4.7.2 Norte Chico

Figura 22. Tres de la tarde en mapa generado, Norte Chico.

Figura 23. Siete de la tarde en mapa generado, Norte Chico.

4.7.3 Zona Centro

Figura 24. Ocho de la mañana en mapa generado, Zona Centro.

Figura 25. Siete de la tarde en mapa generado, Zona Centro.

4.7.4 Zona Sur

Figura 26. Ocho de la mañana en mapa generado, Zona Sur.

Figura 27. Nueve de la noche en mapa generado, Zona Sur.

4.7.5 Zona Austral

Figura 28. Ocho de la mañana en mapa generado, Zona Austral.

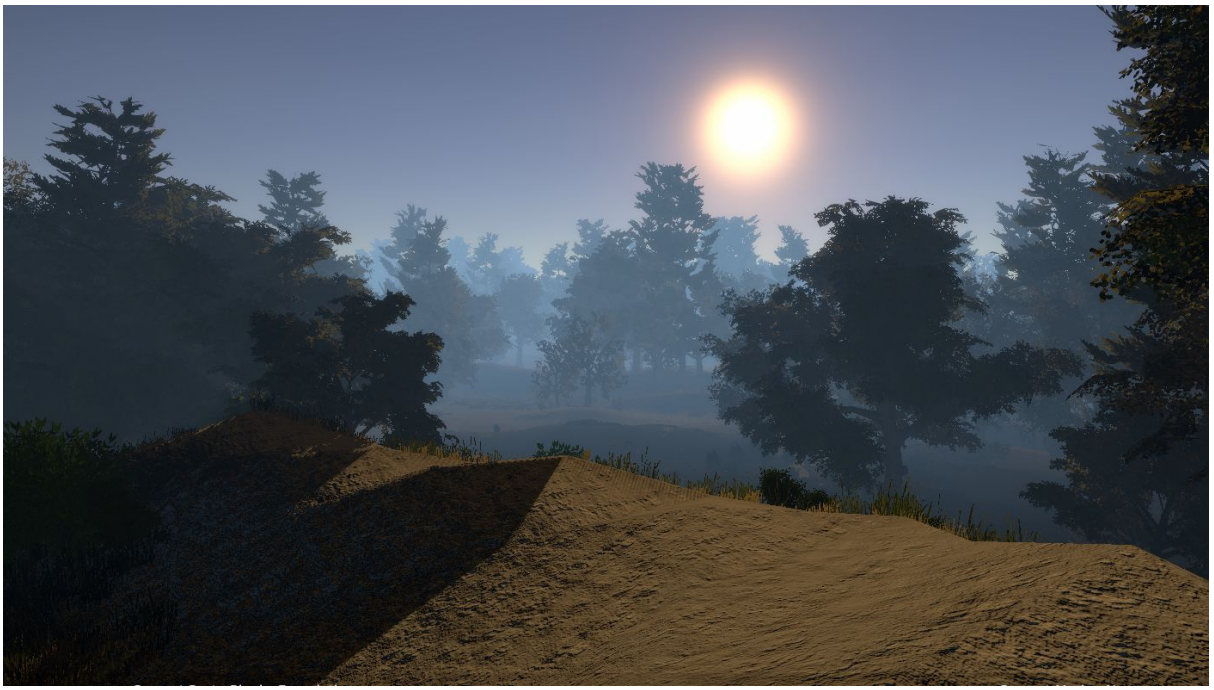

Figura 29. Siete de la tarde en mapa generado, Zona Austral.

4.7.6 GUI

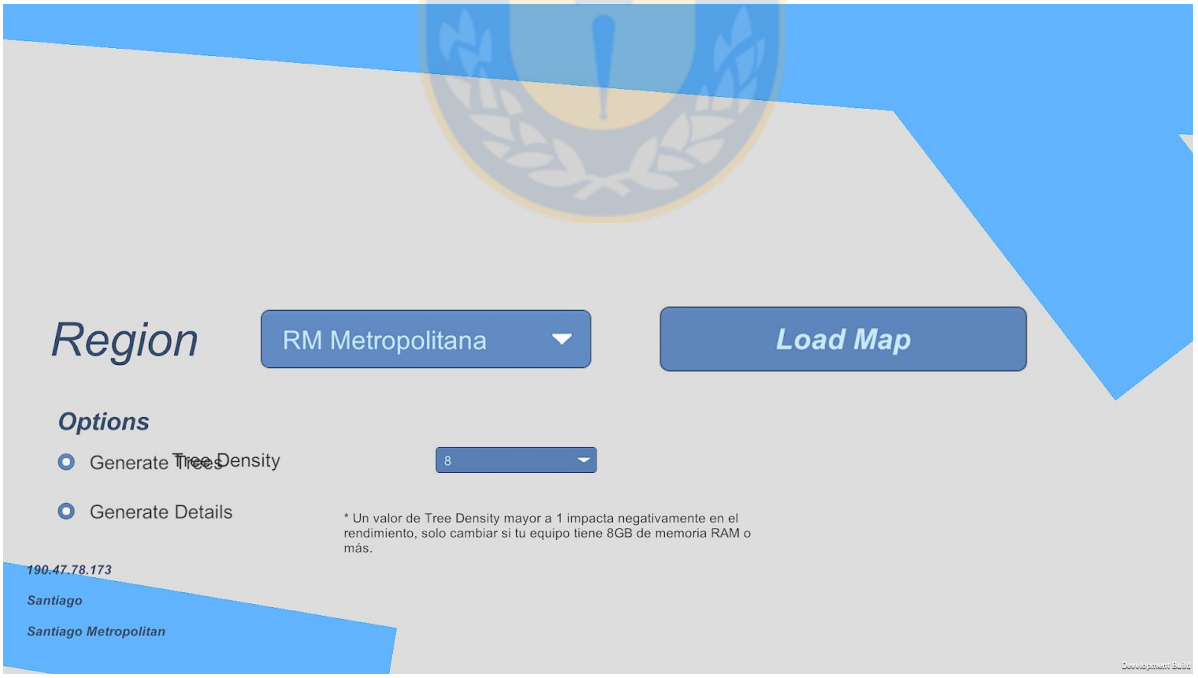

Figura 30. UI del menú principal.

Figura 31. UI del menú de cambio de hora.

Figura 32. UI del sistema de construcción Easy Build System.

5. Evaluación

La evaluación de la plataforma se llevó a cabo con el fin de realizar una validación del prototipo desarrollado bajo los enfoques expuestos y medir la experiencia de usuario, es importante recolectar el feedback de jugadores, ya que ayudaría a guiar modificaciones del proyecto o a desarrollos futuros que tengan en mente la premisa de generación de contenido en base a ubicación.

5.1 Metodología

La etapa de pruebas con usuarios consiste de dos temas principales: pruebas con el prototipo con ayuda de una guía que explica aspectos del prototipo, alcances, controles metodología de testing y contestar una encuesta relacionada a la evaluación del prototipo y a la experiencia como usuario al verse frente a contenido de este tipo.

5.2 Prototipo de pruebas

El prototipo utilizado para las pruebas con usuarios corresponde a un ejecutable de lo trabajado durante el proyecto durante la etapa de desarrollo en el motor Unity con uso de la API de Overpass, la API ip data y los heightmaps generados en base a los datos extraídos desde la SRTM y la IDE Chile. Además, la parte jugable del prototipo corresponde a un sistema de construcción proporcionado por el paquete publicado en la Asset Store de Unity **Easy Build System** [45].

El prototipo está constituido de tres escenarios principales:

1. Escena de obtención de datos de ubicación del jugador.
2. Escena del menú principal, donde por defecto se selecciona la región en la que se encuentra el usuario, pero es posible seleccionar otras regiones y cambiar opciones de generación, como por ejemplo si se desea o no generar árboles y detalles y nivel de densidad de árboles.
3. Escena de generación de escenario según la región y parámetros escogidos.

Aclaraciones sobre el prototipo

1. Los datos obtenidos de la ubicación del jugador son los dados por la consulta a IP API. No son una ubicación exacta, si no que se aproxima según el proveedor de ISP

y toma el nombre de la ciudad o pueblo, región, latitud, longitud e IP. Puede ocurrir que la API consultada devuelva ubicaciones incorrectas, por lo que en ese caso se puede recurrir a elegir el escenario correspondiente de forma manual.

2. Del menú principal no es posible seleccionar la región de Magallanes debido a inconsistencias entre las dimensiones del heightmap de la región y las proporciones de los puntos devueltos por la consulta de overpass.
3. La generación puede tomar alrededor de 2 a 3 minutos. Cualquier cambio hecho durante la partida se perderá al volver al menú principal o al salir del juego.

- Ningún dato personal es recolectado al usar el prototipo, los datos son de uso interno, exclusivo para el funcionamiento del prototipo.

Todo lo descrito anteriormente (y más) está explicado en un documento guía para la etapa de testing.

5.3 Guía del Prototipo y Pruebas

Con el fin de que los usuarios pudieran realizar las pruebas en las mejores condiciones, se creó una guía de usuario para explicar distintos aspectos de la etapa de testing incluyendo imágenes para facilitar su entendimiento. En este documento, se describe el prototipo, sus características, alcances, requisitos técnicos y uso de datos personales, además de la metodología de prueba que debe seguir el usuario con tal de cubrir todos los aspectos que se pretenden evaluar dentro de la encuesta.

A modo resumen, se detallan a continuación los componentes de la guía de usuario para la etapa de pruebas.

5.3.1 Estructura del documento

- Introducción de la guía.
- Requisitos de sistema.
- Prototipo de pruebas y Consideraciones
- Descripción de los componentes del prototipo.
    - Resumen de los componentes
    - Ciclo Día Noche.
    - Sistema de Construcción
        - Modo Emplazamiento.

        - Modo Edición.
        - Modo Destrucción.
        - Piezas de Construcción.
        - Información importante (Tips).
- Controles de juego.
- Metodología de pruebas.
- Evaluación.
- Confidencialidad de los datos.

Se puede ver la versión completa del documento en el enlace del anexo [10.3].

5.4 Encuesta de Experiencia de Usuario y Evaluación del Prototipo

La encuesta tiene como finalidad recolectar información de la experiencia de los usuarios al verse frente a escenarios generados de forma procedimental utilizando como base la propia ubicación. Está dirigido tanto a jugadores como desarrolladores, por lo que cualquiera con conocimientos básicos de videojuegos puede participar. Se busca capturar tanto opiniones positivas como negativas para así guiar desarrollos futuros que utilicen este concepto para apoyar el la creación de productos para distintas industrias, no solo en la producción de videojuegos.

El formulario fue creado en **Google Forms** y contiene distintas secciones para evaluar la experiencia, los métodos de generación implementados, opiniones y comentarios adicionales.

Dentro de lo que se incluye en la encuesta, se encuentran preguntas relacionadas al reconocimiento de lugares familiares relativos a la geografía de las regiones, como volcanes, cerros, lagos, etc., utilidad y variedad de los marcadores de puntos de interés, reconocimiento y calificación de zonas naturales, etc.

La encuesta se mantuvo activa durante el periodo comprendido entre el **18 de diciembre** de 2019 hasta el **30 de diciembre** (inclusive) de 2019.

5.4.1 Despliegue y Difusión

El despliegue será realizado por medio de Google Drive, donde habrá una carpeta comprimida con el ejecutable del prototipo, un manual de uso para explicar alcances del proyecto, los controles de juego y aclaraciones y un enlace para contestar el formulario.

La difusión de la etapa de pruebas es realizada tanto por medios oficiales como extraoficiales, es decir, dentro de las formas de difundir el esta etapa se incluye:

- A través del jefe de carrera hacia los estudiantes de Ingeniería Civil Informática y/o Facultad de Ingeniería en general.
- A través del profesor encargado del taller de Videojuegos Jorge López hacia los estudiantes de la asignatura.
- Grupos de Facebook relacionados al tema de videojuegos.
- Personas dentro del propio círculo.
- Difusión a través de los testers hacia terceros.

5.5 Resultados de la Evaluación

A continuación, se presenta un resumen de los resultados obtenidos durante la etapa de testing. Durante el periodo de pruebas se obtuvo un total del **15** respuestas.

Debido a que el proceso de pruebas no basta con tan solo responder el cuestionario, ya que se necesita que el tester lea un documento guía, dedique tiempo a recorrer los escenarios generados y sumado al tiempo que tarda cada generación del escenario, esto crea la situación que a no a todos los usuarios contactados les interese participar, por lo tanto, la expectativa de cantidad de respuesta debe ser baja, pero suficientemente alta para poder estimar tendencias en cuanto a la calificación de los diferentes aspectos del prototipo y la experiencia con éste.

Dada la naturaleza de las preguntas, se debe realizar una separación en el tipo de respuestas. Por un lado se tienen las respuestas que se pueden cuantificar en diferentes escalas y, por otro, las preguntas abiertas que corresponden a respuestas sobre la experiencia de juego, opiniones y comentarios.

5.5.1 Respuestas abiertas

En esta sección se presenta de forma ordenada las respuestas de las preguntas de tipo respuesta larga. Cabe mencionar que se realizaron algunas correcciones de tipo gramatical y ortográfica a las respuestas en algunos casos. También puede visualizar el resumen de respuestas en el enlace adjunto al anexo [10.4]

- Describa su reacción al poder jugar algún videojuego en ambientes basados en la ubicación.
    1. *"Juegos basados en realidad o lugares reales en este caso da una sensacion de inmersion para mi en este caso, me permite crear o ajustarme y sentirme más cómodo al realizar actividades dentro del juego."*
    2. *"Se hace grato, dado que le entrega más realismo a la experiencia virtual. Personalmente este punto es fundamental a la hora de elegir un videojuego."*
    3. *"El concepto es bueno, pero creo que lo único que rescaté de mi ubicación era el nombre."*
    4. *"Interesante y creativo."*
    5. *"Tiene un potencial tremendo para juegos de mundo abierto (exploración) o educativos."*
    6. *"Interesante, por ver cómo se interpreta la geografía en un videojuego."*
    7. *"Sorpresa e interés por el desarrollo del proyecto."*
    8. *"Muchas oportunidades de entender la ubicación y la geografía de lugares y su comparación."*
    9. *"Impresionante, ya que se requiere bastante interpretación de datos que no son tan fáciles de interpretar correctamente en un motor gráfico."*
    10. *"Emocionado al poder ser capaz de simular lugares en los que pueda ver estado y que sea vean reflejados en la pantalla como parte del escenario"*
    11. *"Buenisimo, altoque me logre ubicar y reconocer lugares familiares que suelo visitar."*
    12. *"En el momento en el que probé el juego estaba en Calama y me posicionó en Santiago."*
    13. *"Me despierta interés, ya que una experiencia así, fuera de la que ya ofrecen juegos de realidad aumentada puede llevar a que se formen comunidades de jugadores en una cierta localidad."*
    14. *"Bastante original la idea, la función captó mi interés de inmediato."*

15. *“Es un concepto interesante, la implementación de este tipo de tecnologías podría cambiar la forma de crear ambientes y generar atraer al usuario, para que conozca lo distinto que puede llegar a ser un punto real del mundo vs otro, dependiendo la zona (desiertos o grandes planicies vacías) quizás no se encuentre un gran atractivo porque será en su mayoría todo igual, pero es un concepto explorable y que en cierta medida logra llamar la atención, y hay varios estilos de videojuegos que pueden verse beneficiados por este tipo de tecnologías.”*

- ¿Que usos cree que podría tener un producto que utilice la ubicación del usuario como base para generar escenarios? (No solo videojuegos, Opcional).
    1. *“Podría usarse como simulador para construcciones, pruebas de tierra, como un simulador de conducción en ciudades más detalladas a futuro.”*
    2. *“Facilitar la búsqueda de sitios cercanos de interés (tiendas, restaurantes, puntos turísticos, etc.).”*
    3. *“Idear estrategias militares de forma virtual, simular desastres naturales como derrumbes, etc.”*
    4. *“Dar a conocer espacios geográficos y ubicación espacial con varios fines.”*
    5. *“Simuladores de Conducción, enseñanza de la geografía del país con uso de VR, atlas interactivo, integración de escenario con google maps para dar indicaciones utilizando puntos de interés creados en el escenario (ejemplo si alguien quiere ir a San Pedro desde concepción - 21 de Mayo, que se pueda generar ese espacio y para el usuario en google maps pueda tener mejores referencias, a medida que vaya avanzando, se indique "te toparás con la vega monumental en 5 minutos" y ahí se muestre una proyección generada de la vega monumental en pantalla de google car, y en 3D se diga donde debe seguir la calle").”*
    6. *“Reconocimiento de terrenos, simulación de experimentos.”*
    7. *“Mmm es complicada la pregunta, pero, mi abuela siempre decia que queria visitar lugares (pero no podía por salud) y algo como esto sería ideal para ayudar a esa gente a poder visitar esos lugares.”*
    8. *“Aplicaciones de aprendizaje escolar como por ejemplo en realidad aumentada o virtual sobre historia y geografía, además hacer mapas de posición para estrategias (aplicables a juegos), con realidad virtual , en aplicaciones convencionales podría ser útil para acelerar el procesamiento de ubicaciones , o también para el turismo en aplicaciones afines.”*

9. *“Podría ser usado en turismo para hacer más inmersiva la búsqueda de puntos de interés (comparado con la alternativa actual de utilizar Google Maps o similar).”*
10. *“Creo que los arquitectos podrían darle un buen uso respecto a construcciones futuras que se quieran realizar.”*
11. *“Tours virtuales, reconocimiento de campo, curiosidad, cosas asi.”*

- Comentarios Adicionales (Opcional).
    1. *“Luego de 1 hora y media de prueba, puedo decir que como prototipo está bien, se denotan los puntos de interés y se sabe de inmediato cuando uno cambia de zona. en general es un buen juego como prototipo.”*
    2. *“Muy buen concepto, a mi parecer falta una mejor aplicación.”*
    3. *“Este prototipo es una buena e innovadora idea que se puede pulir más y generar un buen producto para distintas áreas.”*
    4. *“Se podría hacer un juego tipo pokemon go con esteroides.”*
    5. *“Se podría usar la generación del norte para un videojuego de enseñar a sobrevivir en el desierto.”*
    6. *“Poder ir eligiendo entre qué tipos de marcadores mostrar, cómo mostrar solo ciudades o mostrar solo cerros.”*
    7. *“Me parece muy interesante y para ser un prototipo, está bien logrado, no es fácil usar los heightmaps de Chile (en Cities Skylines que igual es Unity es difícil recrear ciudades de chile precisamente por el heightmaps). Asi que bkn. Quizás hay cosas que corregir, como Playa Blanca en Talcahuano? cosas así, como la ciudad es pequeña respecto a la cámara, quizás con solo ciudades más importantes se pueda viajar mejor, ya que muchos puntos no se sabe cual es el más importante (quizás pueda ser dinámico eso, onda si viajo solo marcar puntos importantes, y al quedarme quieto cargue datos locales, etc).*
    8. *Muy buena la idea, a grandes escalas puede ser algo bastante innovador, siendo capaz de generar una simulación real de un lugar a partir sólo de la ubicación.*
    9. *“Esta buenisimo, se lo mostré a mi mamá y un hermano mío, ambos pudieron identificar lugares en san pedro (donde vivo yo), tambien me gusto mucho la diferenciación de los ambientes, onda, antofagasta realmente se ve como un lugar árido debería verse, por los detalles como las piedras y la tierra. Biobío también se veía cómo Biobío tenía que verse, un buen humedal. (...) y me*

gusto el cambio de flora según el lugar donde uno está, ponte tú los distintos árboles y flores. (...) Ah también, una pregunta era sobre qué usos se le podría dar a este prototipo, yo creo que para algo como un simulador de vuelo seria perfecto."

10. *"En general la experiencia es bastante buena para no estar completado el juego , seria bueno que si se desarrolla más, agregar herramientas de deep Learning para generar ambientes de otros países a partir del chileno que es tan variado y que contiene en él ambientes de la mayoría de países del mundo."*
11. *"Interesante prototipo, al construir mi casa y ponerle puerta no supe nunca como abrirla jaja; pasando a otro punto, no tenía idea que habían dos cerros llamados Teta Norte y Teta Sur."*
12. *"Un proyecto bastante interesante y con gran potencial."*

5.5.2 Respuestas cuantificables

Aquí se presentan las respuestas resumidas en distintos tipos de gráficos para cada parte de la encuesta (figura 33 a 50).

A. Experiencia General

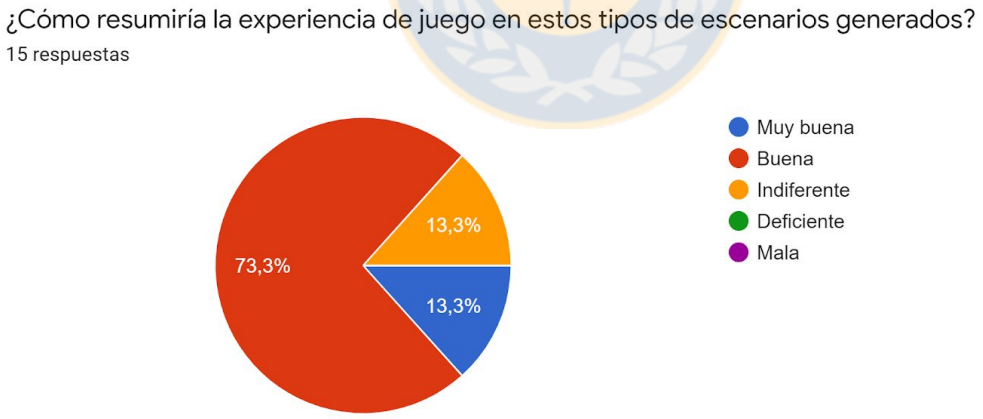

Figura 33. Recuento de respuestas: ¿Cómo resumiría la experiencia de juego en estos tipos de escenarios generados?.

B. Identificación de Geografía

¿Pudo reconocer algunas características geográficas de regiones dentro del juego? (Por ejemplo: montañas, volcanes, bahías, etc.)

15 respuestas

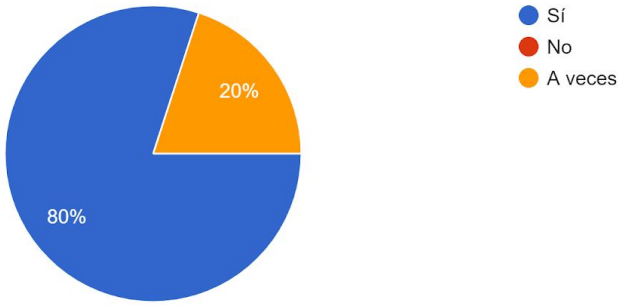

Figura 34. Recuento de respuestas: ¿Pudo reconocer algunas características geográficas de regiones dentro del juego?.

Al comenzar la partida en su región ¿Como describiría la identificación del lugar?

15 respuestas

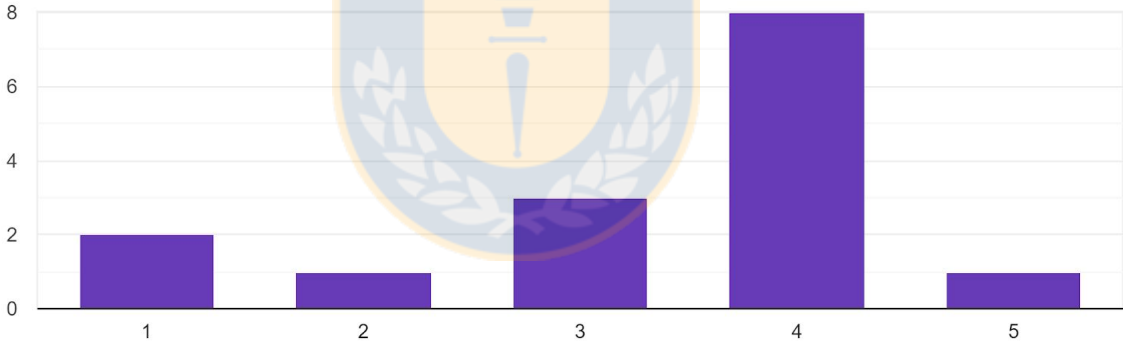

Figura 35. Recuento de respuestas: Al comenzar la partida en su región ¿Cómo describiría la identificación del lugar?.

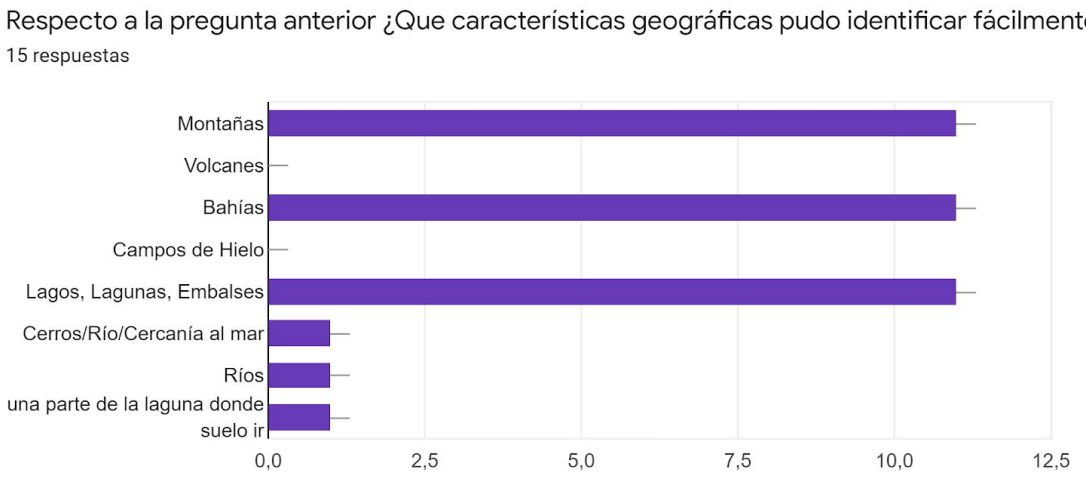

Figura 36. Recuento de respuestas: Respecto a la pregunta anterior ¿Qué características geográficas pudo identificar fácilmente?.

C. Texturizado de Escenarios

Figura 37. Recuento de respuestas: Evalúe en una escala de 1 a 5 la calidad del texturizado de las distintas regiones que se presentan en el prototipo (...).

D. Zonas Naturales

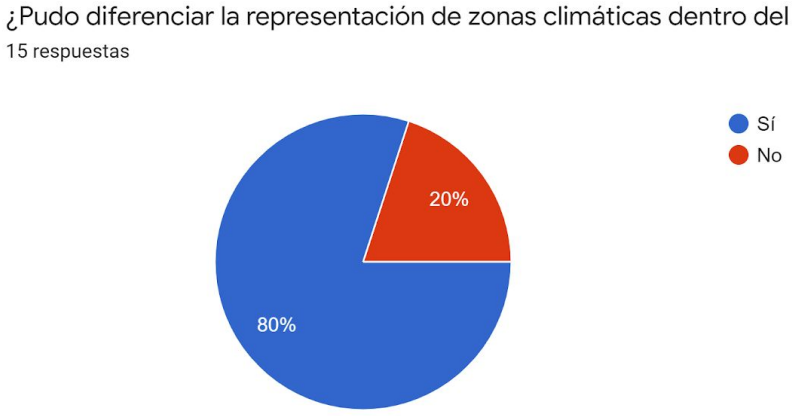

Figura 38. Recuento de respuestas: ¿Pudo diferenciar la representación de zonas naturales dentro del prototipo?.

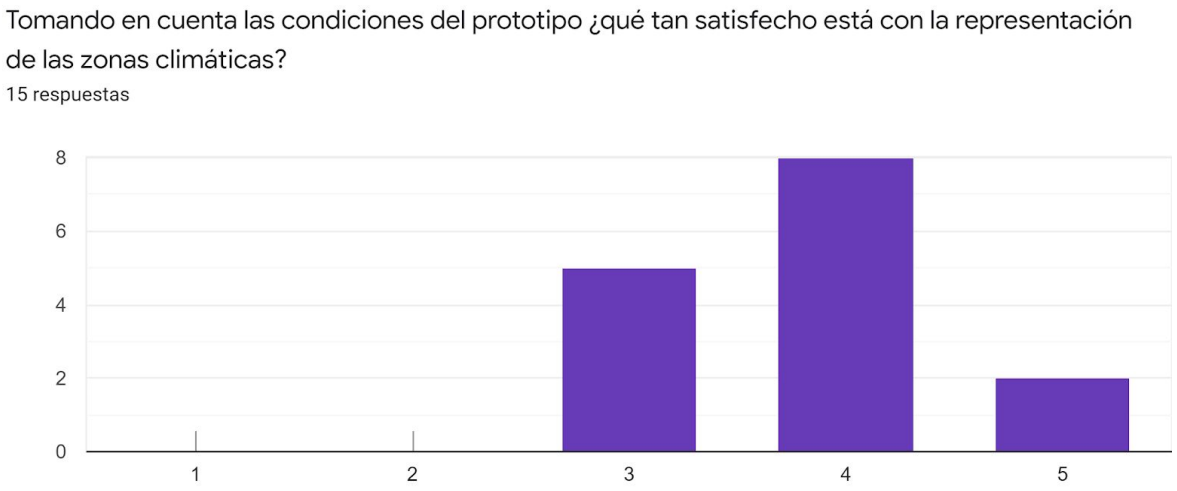

Figura 39. Recuento de respuestas: Tomando en cuenta las condiciones del prototipo ¿qué tan satisfecho está con la representación de las zonas naturales?.

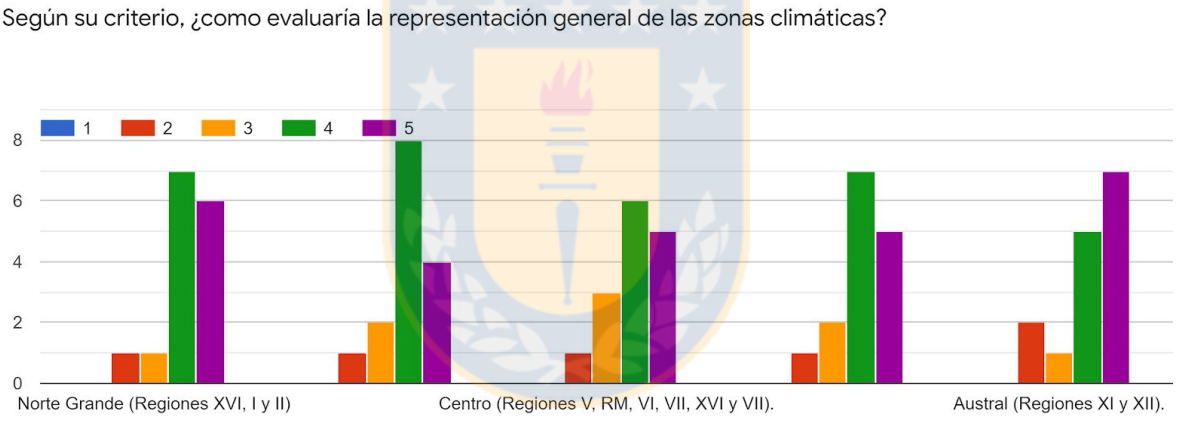

Figura 40. Recuento de respuestas: Según su criterio, ¿cómo evaluaría la representación general de las zonas naturales?.

E. Puntos de Interés

Figura 41. Recuento de respuestas: ¿Ha reconocido puntos de interés sin marcadores en el escenario? (Ejemplos de puntos de interés: ciudades, pueblos, miradores, lagos, cerros, volcanes, etc.).

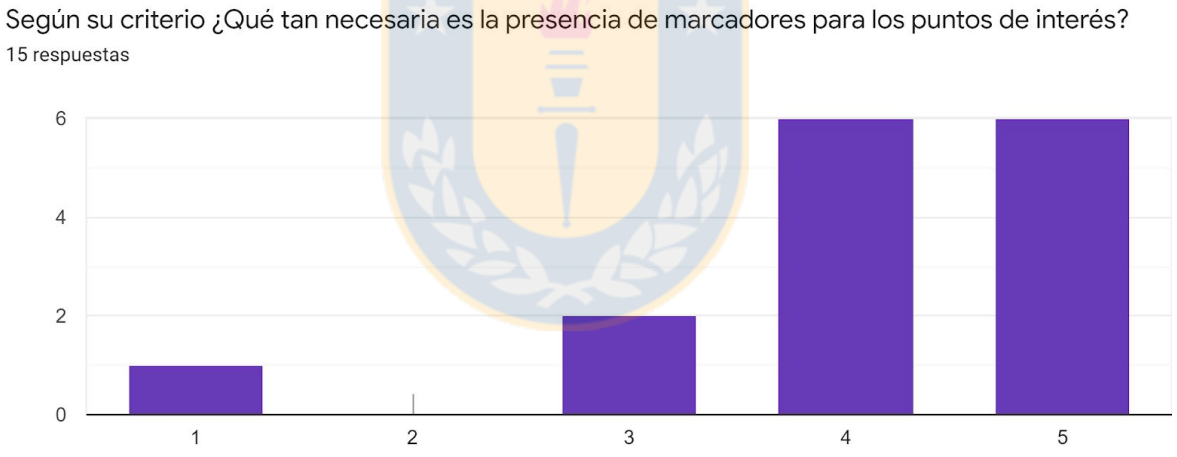

Figura 42. Recuento de respuestas: Según su criterio ¿Qué tan necesaria es la presencia de marcadores para los puntos de interés?.

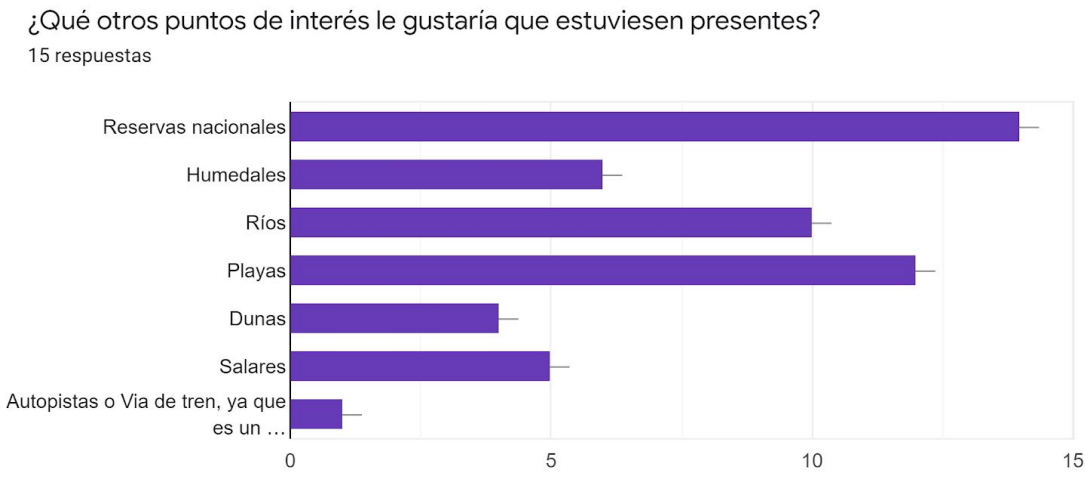

Figura 43. Recuento de respuestas: ¿Qué otros puntos de interés le gustaría que estuviesen presentes?.

***Opciones añadidas por los usuarios**

- Autopistas o Vía de tren, (...).

F. Ciclo Día-Noche Coordinado

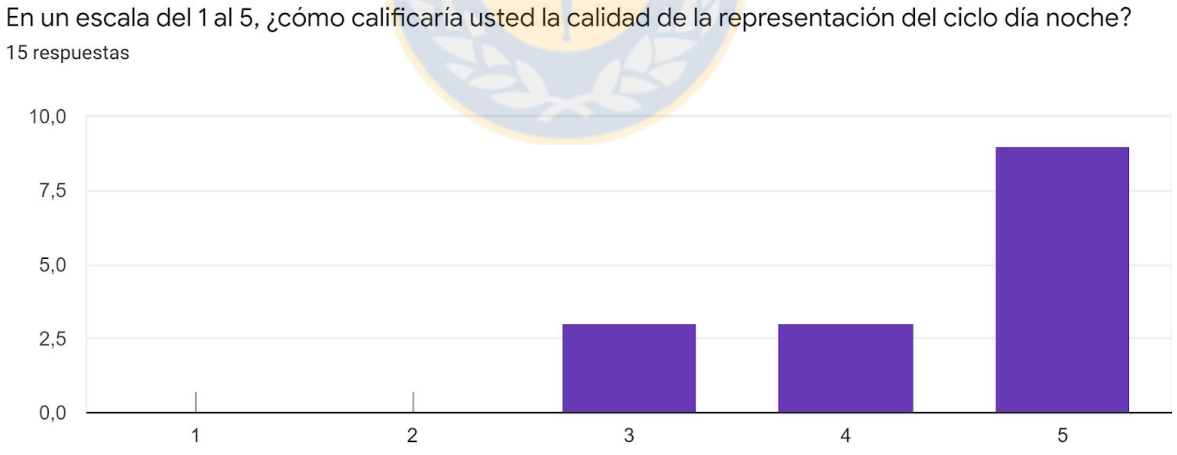

Figura 44. Recuento de respuestas: En un escala del 1 al 5, ¿cómo calificaría usted la calidad de la representación del ciclo día noche?.

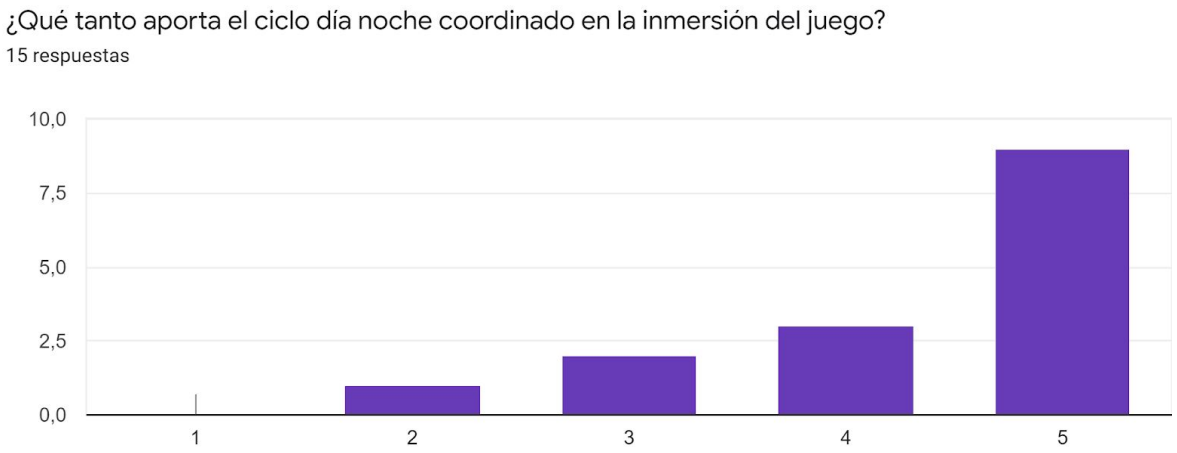

Figura 45. Recuento de respuestas: ¿Qué tanto aporta el ciclo día noche coordinado en la inmersión del juego?.

G. Comparativa

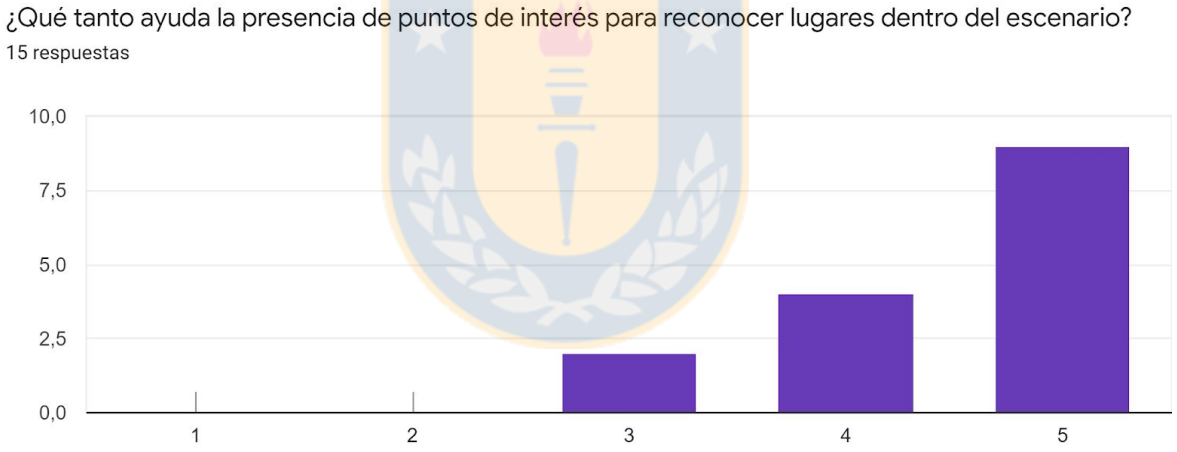

Figura 46. Recuento de respuestas: ¿Qué tanto ayuda la presencia de puntos de interés para reconocer lugares dentro del escenario?.

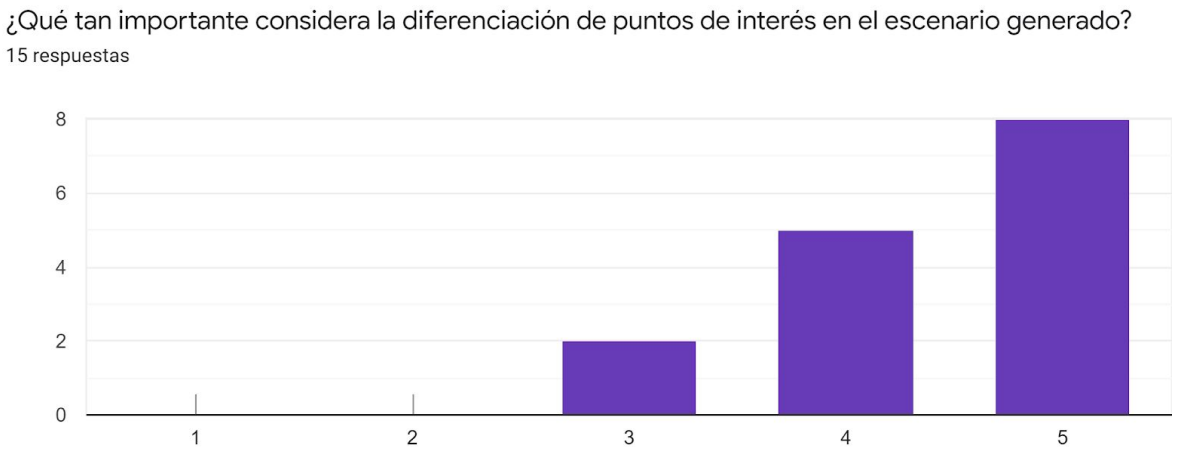

Figura 47. Recuento de respuestas: ¿Qué tan importante considera la diferenciación de puntos de interés en el escenario generado?.

Figura 48. Recuento de respuestas: ¿Que tipos de puntos de interés tienen más relevancia para usted?.

H. Opiniones

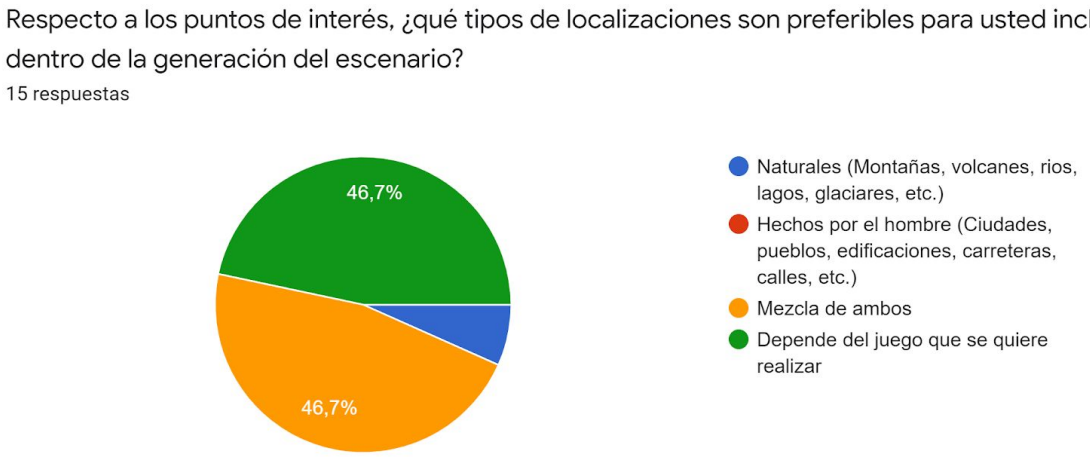

Figura 49. Recuento de respuestas: Respecto a los puntos de interés, ¿qué tipos de localizaciones son preferibles para usted incluir dentro de la generación del escenario?.

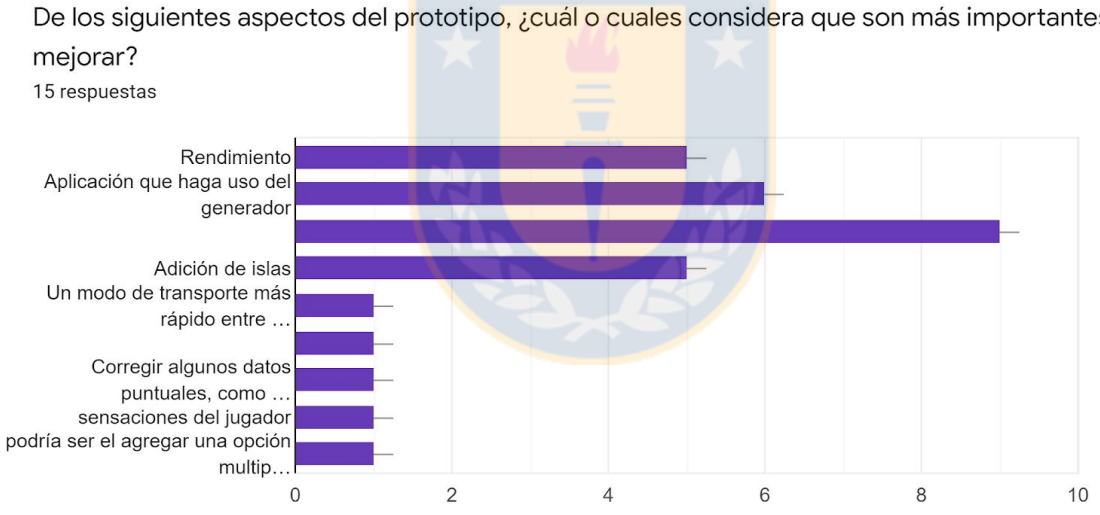

Figura 50. Recuento de respuestas: De los siguientes aspectos del prototipo, ¿cuál o cuáles considera que son más importantes a mejorar?.

***Opciones añadidas por usuarios**

- Un modo de transporte más rápido entre sectores.
- Corregir algunos datos puntuales, como los cactus. Añadir NPC o animales locales como el huemul o cóndores (...).
- Podría ser el agregar una opción multiplayer.

6. Análisis de Respuestas

A partir de las respuestas recibidas se presenta un análisis de la impresión que genera el prototipo creado. De manera análoga a la categorización de los tipos de respuesta de la encuesta, se hará una distinción entre preguntas abiertas y preguntas cuantificables.

6.1 Respuestas a Preguntas Abiertas

- Describa su reacción al poder jugar algún videojuego en ambientes basados en la ubicación.

  A modo general se puede decir que la experiencia de los jugadores fue positiva, destacando el interés por el concepto que propone el proyecto de utilizar simulaciones de lugares para evocar emociones de nostalgia en los jugadores en diversos ámbitos del desarrollo de videojuegos.

  Se puede mencionar la buena impresión que da la diferenciación de zonas naturales y que constantemente está siendo relacionado a un posible uso en educación, principalmente el área de Geografía, o a ciertos géneros o conceptos que se manejan en la industria de videojuegos, como por ejemplo los juegos de mundo abierto o la realidad virtual o aumentada.

- ¿Qué usos cree que podría tener un producto que utilice la ubicación del usuario como base para generar escenarios? (No solo videojuegos).

  Aquí es posible reconocer ciertos patrones en las respuestas que otorga cada jugador. Se menciona reiteradamente su uso para simulación, tanto con fines científicos y educativos, como de entretención y turismo virtual. También se hace mención como una alternativa más interactiva a Google Maps, por ejemplo.

- Comentarios adicionales

  En área de comentarios adicionales se refuerza el interés y el potencial del concepto del proyecto. También se menciona nuevamente el interés por la diferenciación de

zonas naturales y se puede notar que el prototipo puede tener un gran aporte al aprendizaje de geografía nacional.

Se agregan otras ideas para trabajos futuros, como la adición de Deep Learning, videojuegos que se pueden realizar y adición de funcionalidades para visualizar los puntos de interés.

6.2 Respuestas de Preguntas Cuantificables

6.2.1 Experiencia General

En promedio, los usuarios determinaron que la experiencia de juego era buena, es decir, 4 de 5, según lo que muestran los datos obtenidos. Esto demuestra las observaciones realizadas a las respuestas largas sobre experiencia de juego y el interés en el concepto propuesto.

6.2.2 Identificación de geografía

En la mayoría de los casos (**80%**), los encuestados fueron capaces de reconocer distintos hitos geográficos en los escenarios generados. Además, la mayoría de los encuestados considera como relativamente fácil reconocer lugares según hitos geográficos al comenzar una partida, lo que puede tener que ver con el nivel de familiaridad que tengan con los lugares que pueden reconocer dentro del prototipo. En contraste, en algunos casos se da que el reconocimiento de los lugares fue difícil o muy difícil. Esto se puede dar puesto que si una persona no conoce presencialmente un determinado lugar, es muy posible que no pueda reconocer dicho lugar en una simulación virtual de las características del prototipo o que el nivel de detalle de los escenarios simulados no es suficientemente alto para ayudar al reconocimiento.

6.2.3 Texturizado de Escenarios

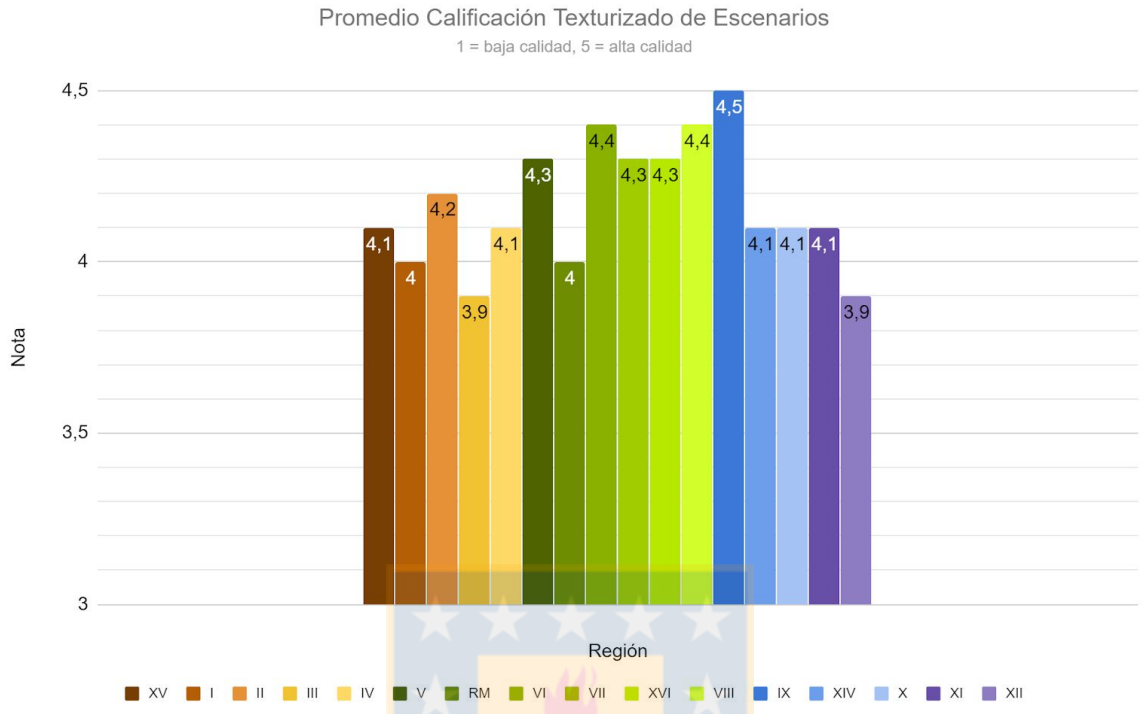

Figura 51. Promedio de calificación de texturizado de escenarios por región.

Como el texturizado de escenarios es evaluado con una calificación de 1 a 5, podemos obtener los promedios de evaluación por región (ver figura 51). De aquí se puede mencionar que la calidad del texturizado fue buena según la percepción de los jugadores, ya que, en promedio la calificación queda en 4.17 puntos a través de todas la regiones.

Si se analiza el texturizado por zona natural, el promedio queda como muestra el siguiente gráfico

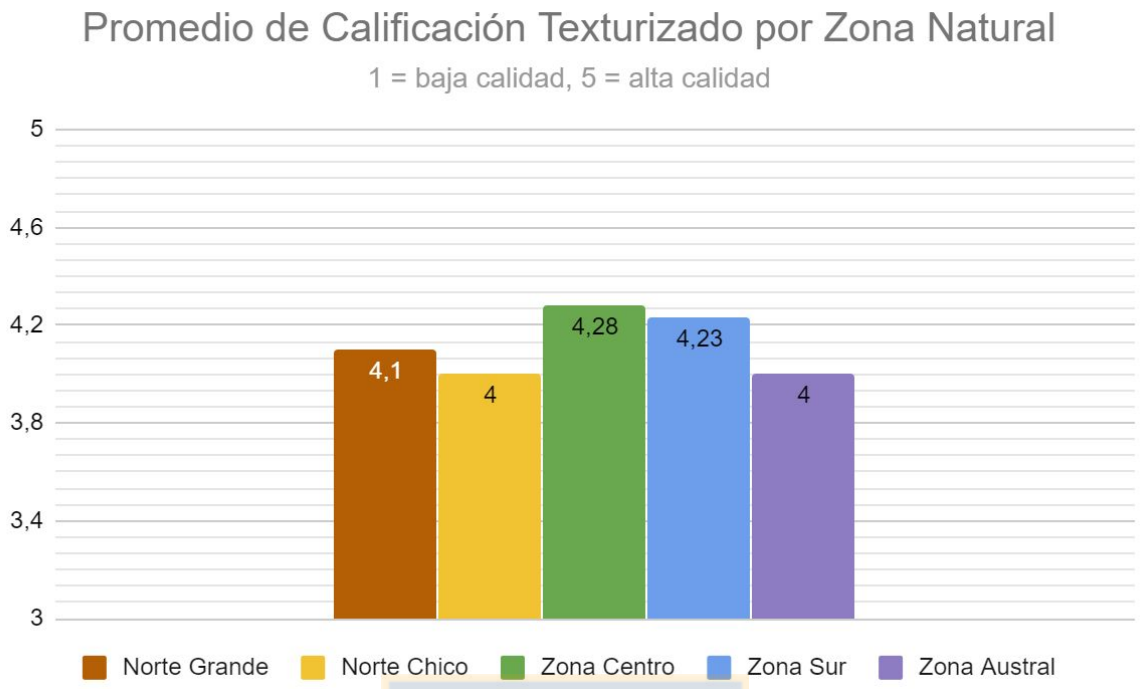

Figura 52. Promedio de calificación de texturizado de escenarios por zona natural.

Si analizamos el promedio por zona, éstas se encuentran bastante equilibradas, sin embargo, la zona con mayor calificación fue la Zona centro, seguida de la Zona sur por un margen muy bajo, luego la Zona del Norte Grande y finalmente, con la misma nota promedio, la Zona del Norte Chico y la Zona Austral (ver figura 52).

Estos resultados refuerzan la afirmación que la calidad del texturizado de escenarios es buena (4) a ojos de los jugadores.

6.2.4 Zonas Naturales

Si consideramos la percepción de los jugadores en cuanto a la diferenciación de las zonas naturales dentro del juego, nos encontramos que un **80%** de ellos fue capaz de diferenciar a simple vista las diferencias entre zonas naturales. Al realizar un análisis de respuestas posteriores, al medir el nivel de satisfacción de la representación de las zonas naturales, la mayoría de los encuestados dice estar satisfecho (**53.3%**), mientras que el resto señala estar indiferentes (**23.3%**) y muy satisfechos (**13.3%**). Aquí se presentó una inclinación a un nivel de satisfacción más bajo, lo que se puede evidenciar que hubo personas que señalaron una discordancia de assets en la representación de algunas zonas naturales (el norte Grande, por ejemplo). Esto se puede justificar por la poca variedad de assets

disponibles (árboles, rocas, detalles, texturas, etc.) durante la fase de desarrollo y algunos desconocimientos sobre las zonas o faltas en la selección de assets para la zonas.

Con fines comparativos, se presenta el gráfico de la calificación promedio de cada zona natural

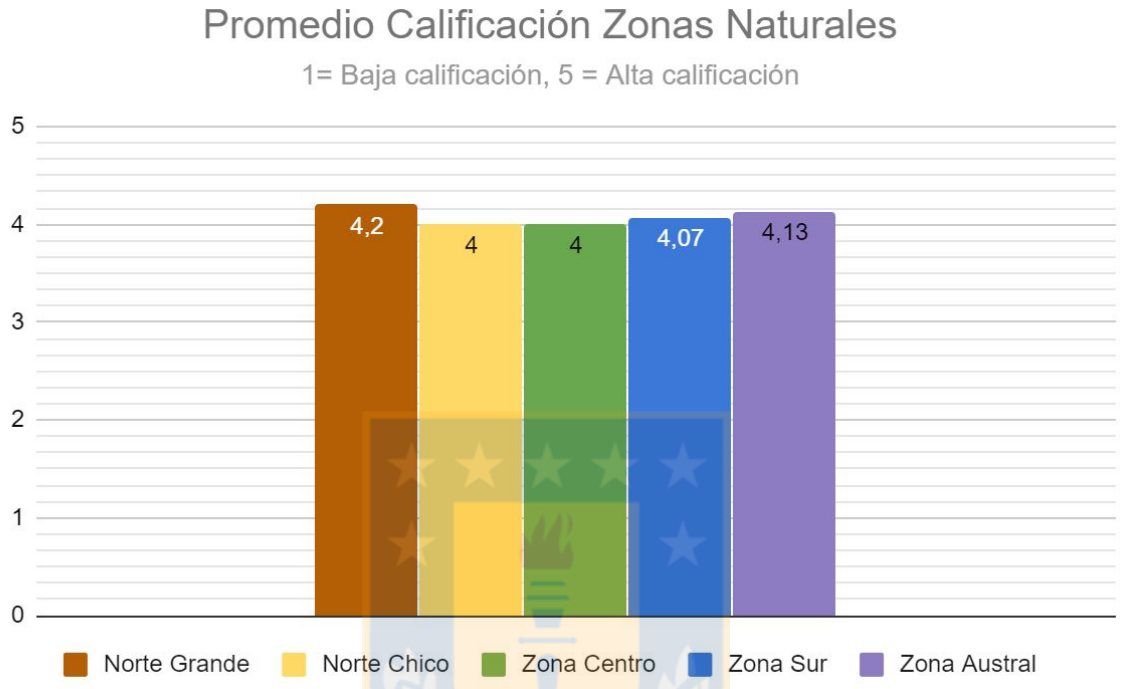

Figura 53. Promedio de calificación zonas naturales.

Si se compara con el gráfico de los promedios de calificación de Texturizado por zona natural (Figura 52), es posible notar que la tanto la Zona Centro (**-0.28** puntos), como la Zona Sur (**-0.16 puntos**) disminuyen su calificación promedio, mientras el Norte Grande (**+0.1** punto) y la Zona Austral (**+0.13** puntos) aumentan su promedio respecto al texturizado y, además, el norte chico sigue igual (figura 53). Como dentro de la categoría de evaluación de zona natural también se incluye el texturizado de escenarios, los aumentos y disminuciones en la puntuación de las zonas se puede deber a la selección de los assets para cada una así como la densidad de árboles, rocas y detalles en cada caso, lo que se puede evidenciar en las sección de opiniones de la encuesta, donde los jugadores señalan que la variedad de objetos en el escenario como punto importante a mejorar. Aun así, en general, la representación de zonas naturales es calificada como buena (**4**).

6.2.5 Puntos de Interés

Dependiendo del jugador, hay diferencias en cuanto al reconocimiento de puntos de interés dentro del juego. Particularmente, se pregunta si es capaz de reconocer algunos puntos de interés relacionados a geografía que no estén indicados con marcadores. Cerca de la mitad de los encuestados (46.7%) asegura que en general pudo reconocer los puntos de interés sin la ayuda de los marcadores, 33.3% no pudo lograrlo y 20% pudo hacerlo en algunos casos. Relacionada a esta pregunta, se pregunta sobre la utilidad de los marcadores de los puntos de interés para ayudar al reconocimiento de lugares o hitos y la mayoría de las respuestas se inclinan por una **gran importancia** de la presencia de estos assets (**80%** de la respuestas le dieron importancia **mayor o igual a 4**). Al probar el prototipo las razones son evidentes: como se trata de una simulación de entornos según la ubicación y las características básicas de cada zona natural, sumado a una relativamente baja precisión de los datos geográficos utilizados (heightmaps), hay hitos y lugares que no son fácilmente reconocibles a menos que se esté familiarizado con el lugar donde se encuentran en la vida real.

Si consideramos otros puntos de interés relacionados a la geografía, los encuestados muestran variadas preferencias. La mayoría de los encuestados señala como importante la adición de **Reservas Nacionales**, **Playas** y **Ríos** dentro del repertorio de puntos de interés, pero en general una mayor variedad de puntos de interés naturales (o relacionados a la naturaleza) son deseables.

6.2.6 Ciclo Día Noche Coordinado

Respecto al ciclo día noche, la mayoría de los jugadores (**60%**) señala como muy buena la representación de los ciclos día noche. La impresión promedio queda en una calidad buena (nota **4**). El aporte de esta característica a la inmersión del juego es significativa y puede ser un punto importante a manejar y mejorar en un futuro.

6.2.7 Comparativa

Con el fin de obtener un balance en las opiniones sobre decisiones de desarrollo, se realizan comparaciones entre características que ofrece el prototipo y otras que no fueron implementadas para lograr identificar lugares o para tener en consideración.

A modo de verificación, para reforzar la idea de la presencia de puntos de interés al preguntar específicamente para reconocer lugares, en lugar de un aporte general a la experiencia, los encuestados, en su mayoría (**60%** califica de **Mucha ayuda**) reafirman la utilidad de esta característica, mientras que el resto otorga menor importancia por el nivel de familiaridad que tengan con los lugares analizados.

En el prototipo, los puntos de interés están diferenciados según el tag por el cual fueron consultados, es decir, las ciudades tienen un gran marcador rojo con muchas construcciones a su alrededor, los pueblos tienen un marcador celeste de menor tamaño con un menor número de construcciones a su alrededor, etc. Esta diferenciación ayudó en mayor medida a dar cierta guía a los jugadores y presentar un entorno más diverso que sólo presentar marcadores de un color en específico de forma general, facilitando la distinción entre puntos de interés.

Como el concepto del prototipo está más centrado en recrear entornos naturales, se consultó a los jugadores sobre sus preferencias en cuanto a tipos de puntos de interés. Los tipos de puntos de interés más preferidos por los encuestados en orden descendente son las ciudades (**86,7%**), cerros, montañas y volcanes (**80%**), lagos y lagunas (**73,3%**), pueblos (**60%**), poblados (**46.7%**) y turismo como campamentos, miradores, etc. (**40%**). En general, comparando con los resultados de los puntos de interés que gustaría que estuviesen presentes, se ve una tendencia en la preferencia por incluir entornos naturales, pero sin olvidar los asentamientos urbanos, dejándolos como pilar principal, ya que forman parte importante de la vida de las personas. Esto se evidencia en la sección de opiniones, que a modo más general se pregunta sobre los tipos de ubicaciones preferibles en cuanto a su origen, si son hechos por el hombre o de origen natural. De aquí se resumen esa preferencia por la mezcla entre lo natural y lo artificial, así como de basarlo según el videojuego / producto que se desea realizar. Sobre esto se basan las ideas de las posibles aplicaciones que se podrían realizar a partir del concepto presentado.

6.2.8 Otras opiniones

Otras opiniones que se midieron fueron los aspectos a mejorar del prototipo o adiciones interesantes que se podrían llevar a cabo. La pregunta admite a adición de opciones por parte de los usuarios para ampliar el feedback, pero aún así, ciertos temas resaltaron. La variedad de elementos en el escenario acorde a la región correspondiente, es el aspecto más importante (**60%**), ya que según los resultados anteriores, el hecho que exista una

amplia variedad de vegetación, objetos, ríos, lagunas, etc. aumenta la facilidad con que se reconocen los entornos generados, mejorando la experiencia en general.

A modo resumen, la impresión general por parte de los encuestados sobre la experiencia con el prototipo y, a más grandes rasgos, el concepto que se propone es **positiva** y en numerosas situaciones se puede deducir que la experiencia con el prototipo aumenta si el nivel de familiaridad con los lugares es mayor, es decir los lugares son conocidos, sumado a la presencia de los puntos de interés como referencia para un reconocimiento más fácil. Se menciona en ocasiones algunos puntos a trabajar en cuanto a la variedad de objetos y vegetación en los entornos, rendimiento en cuanto FPS y tiempo de generación, que es lo que requiere más tiempo. Por otro lado, se sugiere posibles implementaciones de este sistema adicionales a videojuegos, como educación, turismo, arquitectura AR / VR, etc. y mejorarlo en base a conceptos de inteligencia artificial como deep learning.

7. Conclusiones

A partir del prototipo producido y las pruebas con usuarios realizadas durante el transcurso del proyecto, se pudo obtener la información necesaria para verificar el cumplimiento de los objetivos planteados al comienzo de este trabajo.

Se creó como prototipo un juego que hace uso de datos geográficos, ubicación del jugador y algoritmos de procedural generation para crear distintos escenarios que simulan las regiones de Chile, considerando la zona natural a la cual pertenecen, exponiendo a los jugadores a experimentar un juego que corresponda a lugares conocidos, apelando a la familiaridad que tenga el usuario con estos sitios. Los resultados de las encuestas revelaron los jugadores pudieron reconocer lugares dentro del juego los cuales visitaron alguna vez y que la familiaridad con aquellos lugares es un concepto que genera interés en los jugadores. Es por esto que se cumple el objetivo general de este trabajo.

Teniendo en cuenta los objetivos específicos planteados, estos fueron cumplidos, ya que se logró crear el prototipo con integración de geolocalización, aunque no tan exacta como se esperaba, ya que se encontraron ciertas inconsistencias al depender de la localización por medio de IP que alteran la ubicación devuelta por los servidores. Por otro lado, si se consideran las técnicas de generación utilizadas para producir los escenarios, se utilizaron heightmaps para la geometría de terreno, slope maps y flow maps para apoyar el texturizado, generación de procedural meshes a partir de un conjunto de puntos entregados por Overpass (OSM), componentes aleatorios para la distribución de vegetación y detalles de terreno, la implementación de un sistema de ciclo día noche. Todas las técnicas mencionadas anteriormente son parte de la generación procedimental que pueden ser mejoradas en un futuro.

Finalmente, con ayuda de la encuesta y la etapa de pruebas con jugadores, se pudieron analizar las distintas experiencias y opiniones de los jugadores al utilizar el prototipo creado, generando respuestas bastante positivas (en su mayoría), de interés y apoyo para el concepto, el proyecto y posibles trabajos futuros para enriquecer experiencias de los usuarios y aportar en otras áreas ajenas al desarrollo de videojuegos, como turismo, arquitectura, educación, simulación, etc.

El prototipo tiene muchas áreas en las cuales mejorar para poder alzarse como una herramienta alternativa a otras que ya se encuentran en el mercado en cuanto a generación de escenarios, pero el concepto que sugiere puede ayudar enormemente a dar nuevas experiencias a los jugadores apelando a la nostalgia y familiaridad con su entorno, a la vez que se logra disminuir tiempos de desarrollo y costos para estudios que se vean limitados en ese ámbito.

8. Trabajo a Futuro

Desde las etapas finales del desarrollo del prototipo se pensó en formas de cómo mejorarlo considerando las propias ideas y las opiniones de los usuarios. Si observamos otras soluciones de generación de escenarios que ya se encuentran en el mercado, la diferencia es muy grande en cuanto a características.

Diversos aspectos pueden ser trabajados tanto para mejorar los escenarios generados, como la experiencia de los usuarios, dependiendo del enfoque que se le quiera dar a la herramienta, dentro de los cuales se puede destacar:

8.1 Rendimiento

- Tiempo de generación.
  Según un tercio de los encuestados, uno de los aspectos a mejorar del prototipo era el rendimiento, en cuanto a cantidad de FPS y tiempos de carga. Tomando como inspiración los trabajos realizados por los desarrolladores de herramientas de generación ya existentes, debido a la naturaleza de la generación es posible disminuir drásticamente los tiempos de generación de los escenarios utilizando técnicas de paralelismo tanto en CPU como en GPU. Se exploró la opción de utilizar **Compute Shaders** para relegar a la GPU el procesamiento de las etapas de Texturizado, Ubicación de árboles y capas de detalle, que son las etapas que más tiempo toman en procesar.
- Integración con nuevos sistemas de Unity para una mejor utilización de recursos del procesador, como el sistema **Data-Oriented Technology Stack** (DOTS) [44] que está enfocado en aprovechar más núcleos / hebras del procesador y optimizar el uso de la caché.

8.2 Calidad visual

- Aprovechar el potencial del sistema de terrenos de Unity.
  Se puede extender el tamaño del terreno y la calidad visual lograda utilizando un terreno generado por tiles de menor tamaño con mayor nivel de detalle aprovechando todo el potencial del sistema de terrenos de Unity. Esto va de la mano directamente con la generación acelerada por GPU, puesto a que sería necesario procesar una mayor cantidad de datos si se aumenta la calidad y tamaño de los entornos.
- Utilizar un sistema de terrenos de creación propia que permita mayor control sobre sus componentes.
- Utilización de los datos ráster de la SRTM con muestreo de 1 arc-segundo (30 metros) para aumentar el nivel de detalle del terreno, así se podría lograr generación de lugares a más bajo nivel, como por ejemplo, provincias o comunas e incluso ciudades, o derechamente para aumentar el detalle de las regiones.

8.3 Diversidad de entornos

- Aprovechando una disminución de tiempo de la generación utilizando compute shaders para procesar datos en la GPU o utilizando las mejoras del motor (DOTS), se pueden incluir aún más variedades de detalles como plantas, árboles, pasto, rocas, etc. que aporten a la diversidad del entorno. Esto es un punto importante para mejorar la impresión sobre los escenarios generados, para acercarlos más a la realidad.

8.4 Diversidad de estilos

- Al crear un sistema de terrenos propio, se puede extender este sistema para dar soporte a distintos estilos artísticos para la creación de entornos, desde low poly hasta fotorealismo.

8.5 Complejidad de Algoritmos y Sistemas

- Adición y perfeccionamiento de sistemas ya presentes para para enriquecer los entornos. Por ejemplo, se pueden añadir sistemas de clima que reaccionen según el clima predicho para la zona, sistemas de biomas, fauna, etc.

- Se pueden perfeccionar los sistemas ya presentes, como por ejemplo el sistema de ciclos de día y noche con variaciones de la duración de cada fase según la estación del año o añadir otras técnicas de texturizado para enriquecer la variedad del entorno.
- Implementar una generación de ciudades que simule de forma más precisa una generación en base a ubicación.
- Implementar técnicas de manipulación y análisis de imágenes para decorar los escenarios en base a simplificación de colores de una vista aérea de lugares, lo que podría producir entornos más detallados y cercanos a la realidad.

8.6 Aplicación en otras áreas

- De los resultados de las encuestas se despertó el interés por implementar este tipo de sistemas en educación, arquitectura, turismo virtual, simulaciones y ciertos tipos de géneros de videojuegos que se benefician de escenarios de mundo abierto. De esta idea se puede desprender el producir una herramienta de apoyo al desarrollo para motores gráficos que permita en poco tiempo producir escenarios basados en ubicaciones reales con alto margen de modificación.

9. Referencias

1. Dean Takahashi (2016), *Worldwide game industry hits $91 billion in revenues in 2016, with mobile the clear leader*, Venture Beat, 21 de diciembre de 2016. Sitio web: http://venturebeat.com/2016/12/21/worldwide-game-industry-hits-91-billion-in-revenues-in-2016-with-mobile-the-clear-leader/

2. Mario Arias (2017), *Hacer un videojuego es más caro que una peli de Hollywood. 2017*, de urbantecno, Sitio web: https://urbantecno.com/videojuegos/cuanto-cuesta-hacer-videojuego

3. Moss, Richard (1 Enero de 2016). *7 uses of procedural generation that all developers should study*. Gamasutra. Retrieved January 1, 2016, Sitio web: http://www.gamasutra.com/view/news/262869/7_uses_of_procedural_generation_that_all_developers_should_study.php

4. Foro de discusión PUBG (2019), *Tired of playing the same map*, Recuperado de https://forums.pubg.com/topic/327033-tired-of-playing-the-same-map/

5. Usuario de Steam (2015), *Extracto Reseña Abyss Odyssey. 2014*, de Abyss Odyssey Steam Community, Sitio web: https://steamcommunity.com/id/Wasterbull/recommended/255070/

6. Fandom, *Maps of Battlefield: Bad Company 2*, Battlefield: Bad Company 2 Wiki, de Fandom, Sitio web: https://battlefield.fandom.com/wiki/Category:Maps_of_Battlefield:_Bad_Company_2

7. Fandom, *A Vintage Year*, *Hitman Blood Money mission*, Hitman Wiki, de Fandom, Sitio web: https://hitman.fandom.com/wiki/A_Vintage_Year

8. Shaker, N., Togelius, J. and Nelson M. (2016). *Chapter 4 Fractals, noise and agents with applications to landscapes*. En *Procedural Content Generation in Games* (pp.57-71). Switzerland: Springer International Publishing.

9. World Machine Software, LLC. (2019). *World Machine. 2019*, de World Machine Software, LLC Sitio web: https://www.world-machine.com/

10. BiteTheBytes. (2019), *World Creator. 2019*, de BiteTheBytes, Sitio Web: https://www.world-creator.com/

11. Procedural Worlds (2019), *GAIA Pro (2019)*, de Procedural Worlds, Sitio web: http://www.procedural-worlds.com/gaia/

12. Planetside Software LLC (2019), *Terragen 4. 2019*, de Planetside Software LLC, Sitio web: https://planetside.co.uk/

13. Niantic, Inc, *Niantic Labs*, de Niantic, Inc, Sitio web: https://nianticlabs.com/

14. esri Insider (2015), *Localization and Location as a Service (LaaS)*, 8 Diciembre 2015, de esri, Sitio web: https://www.esri.com/about/newsroom/insider/localization-and-location-as-a-service-laas/

15. Mainak Biswas (Enero 2017), *A Guide to Location as a Service (LaaS), 23 Enero 2017*, de IndusNet Technologies, Sitio web: https://www.indusnet.co.in/a-guide-to-location-as-a-service-laas-2/

16. Mapbox (2019), *Company. 2019*, de Mapbox, Sitio web: https://www.mapbox.com/about/company/

17. Mapbox (2019), *Unity Maps SDK (2019),* Mapbox Docs, de Mapbox, Sitio web: https://docs.mapbox.com/unity/maps/overview/

18. Unity Technologies, *Unity Real-Time Development Platform*, de Unity Technologies, Sitio web: https://unity.com/

19. *Terrain Engine*, Unity Manual, Unity Docs (2019), https://docs.unity3d.com/2018.3/Documentation/Manual/script-Terrain.html

20. Wikipedia (2019), *Heightmap. 2019*, a través de Wikipedia, Sitio web: https://en.wikipedia.org/wiki/Heightmap

21. Unity Technologies (2019), *Terrain Class*, Unity Docs Scripting API (2019), Sitio web: https://docs.unity3d.com/ScriptReference/Terrain.html

22. Unity Technologies (2019), *TerrainData*, Unity Docs Scripting API, Sitio web: https://docs.unity3d.com/ScriptReference/Terrain-terrainData.html

23. QGIS, *Acerca de QGIS: Características*, de QGIS, Sitio web: https://qgis.org/es/site/about/index.html

24. GIMP Team, *GIMP - GNU Image Manipulation Program*, de GIMP Team, Sitio web: https://www.gimp.org/

25. Justin Scrawk (2019), Terrain Topology Algorithms. 2019, de Github, Sitio web: https://github.com/Scrawk/Terrain-Topology-Algorithms

26. Corporación de Fomento de la Producción (Corfo) (1950-1962). Geografía económica de Chile. 3 vols. (1.ª edición). Santiago de Chile: Editorial Universitaria, a través de Wikipedia. Sitio web: https://es.wikipedia.org/wiki/Regiones_naturales_de_Chile

27. ip-api.com, *IP Geolocation API*, de ip-api.com, Sitio web: http://ip-api.com.

28. Newtonsoft (2019), *JSON.NET. 2019*, de Newtonsoft, Sitio web: https://www.newtonsoft.com/json

29. Derek Watkins, *SRTM Tile Grabber*, de Derek Watkins, Sitio Web: https://dwtkns.com/srtm/

30. Derek Watkins, 30 Meter *SRTM Tile Downloader*, de Derek Watkins, Sitio Web: https://dwtkns.com/srtm30m/

31. Eric Ramirez, U.S. Releases Enhanced Shuttle Land Elevation Data. 2020, de la NASA, Sitio web: https://www2.jpl.nasa.gov/srtm/

32. *TerrainData.SetAlphamaps*, Unity Docs (2019), https://docs.unity3d.com/ScriptReference/TerrainData.SetAlphamaps.html.

33. *Procedural Terrain Splatmapping in Unity*, Alastair Aitchison blog (2013), https://alastaira.wordpress.com/2013/11/14/procedural-terrain-splatmapping/.

34. Valve Corporation (2019), *Steam Hardware Survey June 2019*, de Valve Corporation, Sitio web: https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam.

35. Unity Technologies (2019), *Textures: Terrain Heightmaps*, Unity Manual, Unity Docs (2019), Sitio web: https://docs.unity3d.com/2018.3/Documentation/Manual/Textures.html#TerrainHeightmaps

36. Unity Technologies (2019), *TreePrototype Class*, Unity Docs Scripting API, Sitio web: https://docs.unity3d.com/ScriptReference/TreePrototype.html

37. Unity Technologies (2019), *DetailPrototype Class*, Unity Docs Scripting API, Sitio web: https://docs.unity3d.com/ScriptReference/DetailPrototype.html

38. Stray Pixels, *Delaunay Triangulation for Terrain Generation in Unity. 2017*, de Stray Pixels (2017), Sitio web: https://straypixels.net/delaunay-triangulation-terrain/

39. Open Street Map, *Map Features. 2019*, de Open Street Map Wiki (2019), Sitio web: https://wiki.openstreetmap.org/wiki/Map_Features.

40. *Triangle.NET*, de Codeplex (2018), Sitio web: https://archive.codeplex.com/?p=triangle

41. Spy Blood Games, *Simple Day and Night Cycle System*, de Unity Asset Store, Sitio web: https://assetstore.unity.com/packages/templates/tutorials/simple-day-and-night-cycle-system-66647

42. One Wheel Studio, *Day Night Cycle Sky Box Module. 2018*, de One Wheel Studio (2018), Sitio web: https://www.youtube.com/watch?v=RIveD6jQ7O8.

43. Narendra, A., Reid, S.F., & Hemmi, J.M. (2010). *The twilight zone: ambient light levels trigger activity in primitive ants. Proceedings. Biological sciences, 277 1687*, 1531-8. Imagen recuperada desde: https://royalsocietypublishing.org/doi/10.1098/rspb.2009.2324.

44. Unity Technologies, *Unity Data Oriented Technology Stack. 2019* , de Unity Technologies, Sitio web: https://unity.com/dots

45. ADS Studio 12, *Easy Build System - Modular Building System*, Unity Asset Store https://assetstore.unity.com/packages/templates/systems/easy-build-system-modular-building-system-45394

46. Khan Academy, *Perlin noise*, Khan Academy, Sitio web: https://www.khanacademy.org/computing/computer-programming/programming-natural-simulations/programming-noise/a/perlin-noise

10. Anexo

10.1 Consultas

A continuación se detallan las consultas utilizadas para extraer la información geográfica para las puntos de interés, polígonos de cuerpos de agua y puntos centrales de las regiones.

1. **Consulta de ejemplo para extraer nodos de interés**

```
[out:json];
area[name="Región del Biobío"]->.searchArea;
(
      node[place="city"](area.searchArea);
      node[place="town"](area.searchArea);
      node[place="hamlet"](area.searchArea);
      node[tourism="camp_site"](area.searchArea);
      node[tourism="viewpoint"](area.searchArea);
      node[natural="peak"](area.searchArea);
      node[natural="glacier"](area.searchArea);
      node[natural="hot_spring"](area.searchArea);
      node[natural="geyser"](area.searchArea);
      node[natural="volcano"](area.searchArea);
);
out body;
```

2. **Consulta de ejemplo que devuelve los cuerpos de agua en una región**

```
[out:json];
area[name="Región del Biobío"]->.searchArea;
(
      way[water](if:t["water"]!="river")(area.searchArea);
      way[natural="water"](if:t["water"]!="river")(area.searchArea);
      rel[water]
            (if:t["water"]=="reservoir")
```

```
            (if:t["water"]=="lake")
            (area.searchArea);
      rel[natural="water"][name~"Lag"](area.searchArea);
      rel[natural="water"][name~"Emb"](area.searchArea);
      rel[natural="water"]
            (if: t["water"!="river"])
            (area.searchArea);
);
out geom;
//out center; (para devolver solo el centro de cada polígono)
```

3. **Consulta que retorna los centros de cada región de Chile**

```
[out:json];
      area[name="Chile"]->.searchArea;
      (
      node[place=\"state\"](area.searchArea);
      );
out body;
>;
out skel qt;
```

10.2 Inconvenientes y Métodos Alternativos

1. **Inconveniente pruebas de heightmap**

   Inicialmente, el tamaño del heightmap era de 4097x4097, pero alojaba demasiada memoria al ser utilizado para la creación del terreno. Al generar el terreno con los datos contenidos en el archivo, Unity consume aproximadamente 11.5 GB de memoria RAM, lo que no es conveniente ya que limita el espectro de ordenadores en los que se pueden realizar pruebas, ya que, según la Steam Hardware Survey de abril de 2019 [34], la cantidad más común de memoria RAM del sistema es de 8 GB. Cabe mencionar que el nivel de detalle del terreno no se vio tan afectado por la disminución de la resolución del heightmap.

Los heightmaps de 16 bits se importan con errores que provocan una duplicación de datos, lo que se observa como cuatro copias de menor tamaño del terreno. La solución es cambiar la precisión de lectura a 8 bits. Esta situación ocurre por la calidad de color desde el origen de los datos es de 8 bits y, aunque sean importados posteriormente a 16 bits por GIMP, la calidad sigue igual. Luego de importar el heightmap, el terreno es renderizado con un efecto de espejo que cambia la orientación del mapa generado. La solución es dejar marcada como verdadero la opción **Invertir Verticalmente**.

2. Inconveniente Consulta Overpass

Dentro de la consulta a Overpass, se definió la siguiente estructura para los nodos:

```
node[place="city"][name](area.a);
```

La línea anterior devuelve todos los nodos dentro del área definida los cuales tengan un nombre y el tag “place” tenga el valor “city”, por lo tanto devuelve todas la ciudades dentro de una región.

La misma estructura se repite para varios tags mencionados anteriormente en la selección

```
node[place="town"][name](area.a); (pueblos)
node[natural="volcano"][name](area.a); (volcanes)
node[natural="peak"][name](area.a); (cerros / montañas)
...
```

El problema radica en el tiempo total de consulta. Cuando se quiere retornar ciudades, pueblos, montañas, volcanes, etc., este tipo de consulta toma aproximadamente 1 minuto y 15 segundos, lo que se añade al tiempo total de generación del escenario y compromete la escalabilidad de la consulta para otros tag que se quieran añadir.

Tras probar modificaciones en la consulta y tratando de mantener la escalabilidad del formato para la admisión de más tags, se llegó a que el causante de tales tiempos de ejecución era la presencia del tag **name** en cada línea. Como aún es

necesario tener nodos con nombre para que sean significativos dentro del escenario, la solución es simple: eliminar “[name]” de cada línea de consulta y obtener todos los nodos, con y sin nombre, para luego descartar aquellos nodos que no tengan nombre a través de código dentro de los scripts. Las líneas de consulta quedan como sigue:

```
node[place="city"](area.a);
node[place="town"](area.a);
...
```

Con esto se evita el overhead a la API Overpass de buscar por los nodos con nombre en cada caso y el tiempo de consulta disminuye de los **1 minuto 15 segundos a solo 6 segundos**, es decir, aproximadamente un **90%** de disminución de tiempo.

3. Complicaciones en Posicionamientos de POIs

Dado que la API de Mapbox no tiene integración con la herramienta de terreno de Unity, la traducción de las coordenadas de latitud y longitud al espacio de Unity no se puede realizar automáticamente y de forma exacta, por lo que hay que hacer pasos adicionales para ubicar los puntos de interés en el mapa.

La API de Mapbox contiene ciertas utilidades de la para para traducir coordenadas esféricas de latitud y longitud a un plano XY (XZ en el caso de Unity, Y es para indicar la altura). Al utilizar las funciones que provee Mapbox, las coordenadas se traducen en valores muy altos, del orden de los cientos de miles (por ejemplo: (-203000,300000)) y muy alejados del centro (0,0) del plano de Unity, que es donde se ubica el terreno. El terreno ocupa el espacio comprendido entre el origen (0,0,0) y el punto (8192,0,8192), por lo tanto hay que traer esos puntos desfasados a la ubicación del terreno para poder realizar el posicionamiento correcto.

El primer método es calcular el punto medio de todos los datos obtenidos desde la consulta de overpass para luego sustraerlo de cada uno, esta solución incurre en cálculos adicionales para obtener el punto medio.

El segundo método es utilizar un “punto medio de referencia” extraído desde los mismos datos que ofrece Open Street Map a través de Overpass (ver figura 54), que fue el método que dio mejores resultados. En este caso se obtienen todos los puntos de los centros de regiones en una consulta a la API de Overpass (ver anexo [10.1, 3]) y se almacenan en cada objeto de región como información adicional.

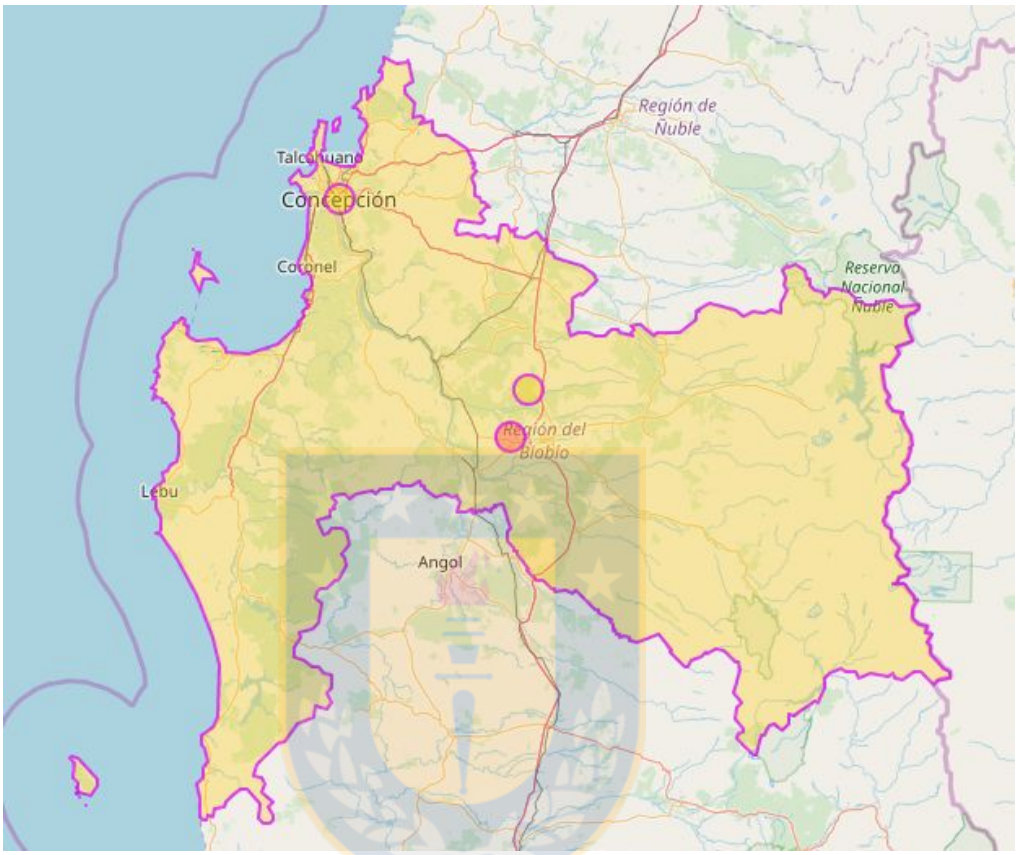

Figura 54. Región del Biobío, Overpass Turbo. El punto más oscuro corresponde al punto de referencia obtenido mediante consulta a la API.

Al obtener los puntos de interés, la traducción usando las funciones de Mapbox se realiza sustrayendo el punto central obtenido al principio de las coordenadas de cada punto de interés, considerando este centro obtenido como centro del mapa y disminuyendo la escala de acercamiento desde 1 hasta un valor dentro del rango [0.01000,0.05000] dependiendo del tamaño de la región seleccionada.

Otra complicación encontrada en esta etapa supuso una disminución de la escala con la que se entregan los datos desde Overpass ya que el valor 1 corresponde a una relación 1:1 entre metros reales (OSM) y unidades de Unity y como el terreno es a tamaño escala mucho menor a la realidad, era necesario disminuir el valor. Luego se realizó un segundo ajuste, ya que los puntos estaban distribuidos a lo largo de

todos los cuadrantes, pues el punto central queda en el origen (0,0,0), mientras que el terreno se encontraba en el primero. Entonces, a cada punto se suma la mitad del tamaño del terreno en cada coordenada, lo que mueve a todos los puntos dentro del primer cuadrante. Sin embargo, la ubicación de los puntos necesita de un **offset** para calzar de forma más precisa, ya que el centro obtenido de la región no queda justo en el centro (4096,0,4096) del terreno. Como las escalas y puntos centrales son distintos entre cada región y a causa de la redimensión de los heightmap de cada una de ellas en el proceso de creación, fue necesario estimar estos valores de forma **manual** para ajustar las posiciones finales dentro de un rango admisible.

4. Métodos previos de localización de cuerpos de agua en el terreno

Método 1

El primer método intentado fue extrayendo el punto determinado como centro del polígono a través de una consulta a Overpass (Ver anexo [10.1, 2]), para luego ubicar un prefab que represente el agua en el lugar del punto por medio de Raycast, similar al acercamiento con los puntos de interés anteriores.

Ventajas

- Ocupa poca cuota diaria de Overpass ya que se devuelven solo nodos.

Desventajas

- No en todos los casos, el centro entregado por la consulta a Overpass estaba dentro del polígono, caso que se daba para cuerpos de agua con una forma cóncava, lo que obligaba a buscar una forma de encontrar el punto donde debería estar el mapa, mientras que el segundo problema era que solo obteniendo el punto central, no se puede determinar el tamaño de el cuerpo de agua, por lo tanto no se puede determinar el tamaño del prefab que debe ser ubicado.

Método 2

El segundo método corresponde a extraer todos los puntos que forman el polígono de cada cuerpo de agua y crear un mesh de forma procedural utilizando estos puntos en Unity.

Un mesh está formado por tres arreglos: vértices, triángulos y coordenadas UV. Los vértices son puntos en tres dimensiones, los triángulos se definen por tres números que corresponden a los índices de los vértices que lo forman en sentido antihorario (en sentido horario, la cara visible del mesh es opuesta) y las coordenadas UV son para realizar el mapeado de texturas al aplicar un material sobre el objeto (no siempre es necesario).

El mesh es generado a partir de un conjunto de puntos en dos dimensiones entregado por la consulta a Overpass (ver anexo [10.1, 2]) y convertido a espacio XYZ utilizando el Algoritmo de Triangulación de Delaunay que provee la librería Triangle.NET [40] (ver figura 55 de ejemplo).

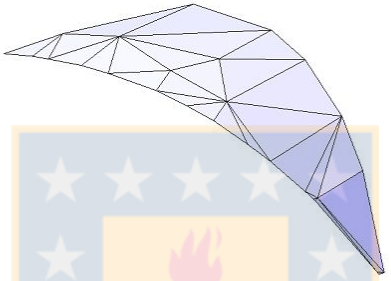

Figura 55. Ejemplo de mesh triangulado.

Ventajas

- Provee la ubicación del cuerpo de agua de forma más fácil.
- La forma del polígono es más ajustada.
- Se previenen bugs ya que la forma es más exacta.

Desventajas

- Usa más datos desde Overpass, lo que puede generar un límite de cuota.
- La triangulación deja varios puntos perimetrales de la figura dentro de ella.
- El posicionamiento aún no es exacto.

5. Inconvenientes de transición día noche

Cuando se llega a la fase de noche, el cielo se torna completamente negro, sin importar el color que le corresponda según la gradiente definida, ya que el sol se

encuentra apuntando hacia la parte inferior del terreno (el sol está debajo del terreno), lo que no da un buen efecto visual.

La solución fue añadir una segunda luz direccional que represente la luna que rote en la misma dirección que el sol pero en sentido opuesto.

Como la procedural skybox utilizada para asignar la iluminación del escenario solo puede representar una luz direccional a la vez, es necesario establecer los tiempos de salida y puesta de sol, para así activar o desactivar cada luz direccional según corresponda.

10.3 Documento Guia de Etapa de Pruebas

Enlace al documento:
https://drive.google.com/open?id=1cvl_q2Zg-08GXCc95sErubY1yC6zXf7M

10.4 Resultados de Encuesta

Enlace al resumen de respuestas:
https://docs.google.com/forms/d/1HMgbxs1rT15FBIrayODz21AEl58ytsdhdIGZ5Cb2yEA/edit?usp=sharing