# Introducción

El presente informe reúne la documentación de la solución del cuarto trabajo practico de la materia **Taller de Programación I** que consiste en desarrollar un remake del videojuego de estrategia Dune en **C++** utilizando los conceptos del paradigma de la programación orientada a objetos vistos hasta ahora en el curso.

# Informe General del Proyecto

## Problema

Consiste en desarrollar un remake del videojuego Dune, multijugador en linea. La aplicación debe soportar al menos un juego para una cantidad arbitraria de jugadores (al menos dos por la jugabilidad).

Dune es un juego de estrategia en tiempo real, en el que cada jugador controla una civilización y debe sobreponerse a los otros jugadores para ganar. Los jugadores pueden crear unidades para pelear o para recolectar especia, necesaria para la economía de la civilización. Cada jugador tiene un centro de construcción, que es el edificio principal de la civilización. Si un jugador pierde su centro de construcción, este pierde y debe abandonar el juego.

## Funcionalidades alcanzadas

1. El servidor recibe un archivo de configuración como parámetro, donde se detallan las opciones de ejecución: El puerto por donde escuchar conexiones, la cantidad de jugadores, el mapa y un archivo .json de propiedades de unidades.
2. En este se detallan todos los valores numéricos correspondientes a las velocidades de las unidades, los puntos de ataque, el costo de creación, etc.
3. El mapa es un archivo .json donde se detalla el terreno (arena, roca, dunas, …) las posiciones de los centro de construcción y de las especias.
4. El servidor espera a que todos los clientes se conecten, luego comienza el juego.
5. En cualquier momento, el servidor termina cuando se ingresa la letra **q** en la consola.
6. Cada jugador comienza con un centro de construcción, 2 cosechadoras y un Trike. Las posiciones de las unidades son relativas al centro de construcción.
7. Cada jugador es diferenciado por el color de sus unidades y edificios.
8. Los jugadores pueden mover sus unidades, recolectar especia y atacar a enemigos usando el mouse.
9. Las unidades atacantes reajustan su posición si el objetivo se desplaza.

10. Si una unidad es atacada, esta responde el fuego automáticamente.
11. Las unidades encuentran automáticamente la trayectoria a un determinado punto en el mapa.
12. Un jugador puede seleccionar a varias unidades a la vez arrastrando el mouse.
13. Cuando un jugador selecciona una unidad, puede ver cuantos puntos de vida le quedan.
14. Las unidades no pueden moverse a través de roca o edificios. Los vehículos no pueden moverse a través de cimas.
15. Las unidades tienen animaciones de movimiento y gráficos que dependen de la dirección.
16. La recolección de especia es realizada por las cosechadoras. Estas se dirigen a la especia, y una vez que se llena su capacidad, van automáticamente al silo mas cercano a guardarla, luego repiten el ciclo. Si no hay un silo, se quedan esperando las instrucciones del jugador.
17. Si se agota la especia, las cosechadoras buscan automáticamente otra cercana.
18. El jugador tiene una capacidad limitada de especia. Para aumentarla se deben construir mas silos o refinerías. Si la cantidad de especia llega al limite del jugador, las cosechadoras se quedan esperando indicaciones.
19. En cualquier momento, las cosechadoras reinician su ciclo enviándolas a una especia.
20. Los jugadores pueden vender sus unidades y edificios recuperando la mitad de su costo.
21. Los jugadores pueden crear edificios usando la interfaz grafica. Crear edificios habilita la creación de unidades.
22. Se pueden crear varios edificios o unidades a la vez, siempre que sean de distinto tipo.
23. Se puede ver el avance de la creación de unidades o edificios.
24. Los edificios se posicionan donde lo indique el jugador usando el mouse. Las unidades se posicionan automáticamente cerca de su edificio de creación.
25. Los jugadores pueden ver que edificios y unidades tienen disponibles para su creación a través de la interfaz grafica.
26. Para crear edificios se necesita energía, esta se obtiene construyendo un tipo especial de edificio llamado trampas de aire.
27. Los jugadores poseen de una vista de águila, que muestra el mapa en su totalidad en una escala mas pequeña.
28. Los jugadores son informados de la cantidad de especia y energía que poseen en todo momento.
29. Un jugador pierde cuando se destruye su centro de construcción. Luego se cierra el juego automáticamente.

30. Un jugador gana cuando es el único en el juego. Luego se cierra el juego automáticamente.

## Limitaciones

1. El servidor no soporta varios juegos a la vez. Se debe reiniciar con cada juego.
2. Los ataques entre unidades no poseen animaciones, ni existen los proyectiles.
3. No existen las unidades especiales Fremen, Sardaukar, Tanque Sonico, Desviador y Devastador.
4. Al no existir las unidades especiales, el edificio Palacio no sirve, por lo tanto fue eliminado.
5. De la misma forma, las casas solo cambian en los gráficos de los cuarteles. Por lo tanto las casas serán elegidas automáticamente por el juego para cada jugador.
6. Las unidades no se atacan automáticamente por cercanía.
7. No hay música, mensajes de voz ni sonidos.
8. No hay animaciones para el ataque o destrucción de unidades.
9. El mapa no se desplaza cuando el jugador mueve el mouse a los bordes de la pantalla.
10. No se llego a completar la implementación del editor.
11. Los mapas no son almacenados en formato YAML, sino en json.

De contar con mas tiempo, se hubiera desarrollado el editor, el auto ataque por cercanía, las animaciones de ataque y destrucción.

# Informe Técnico

## Solución

Son tres aplicaciones, una para el servidor, otra para el cliente y otra para el editor de escenarios.

Al ser un juego en tiempo real, todos los clientes deben actualizar sus movimientos al servidor. Aquí se encuentra el modelo del juego, que se actualiza según los eventos recibidos por el cliente. Una vez procesados los eventos, el servidor envía el estado actual del juego a todos los clientes a un ritmo constante. De esta forma los clientes se mantienen actualizados entre si y todos los cambios son controlados por el servidor. El servidor es una aplicación de consola.

El cliente es una aplicación grafica hecha con SDL. Tiene dos funciones: 1. Recibe el estado del juego actual enviado por el servidor y lo grafica con SDL. 2. Obtiene las acciones del cliente (mouse, teclado, etc), las procesa en forma de eventos y se las envía al servidor para que actualiza el modelo.

# Modelo

## Server

Se modela con clases de C++:

La clases principales del servidor son:

1. `Accepter` escucha conexiones de los clientes y crea un objecto `ClientThread` para cada uno de ellos. Termina cuando llega a la cantidad deseada de jugadores.
2. `ClientThread` es la clase que administra la conexión de un cliente. Corre en su propio thread y su funcion es recibir los mensajes del cliente, convertirlos a eventos y enviarselos a la clase `Game` para que los procese.
3. `Game` comienza su ejecución cuando finaliza `Accepter`. Comienza por enviar las configuraciones iniciales del juego y luego entra a un loop que finaliza cuando termina el juego. En este loop: 1. Se obtienen todos los eventos recibidos por los clientes. 2. A partir de esos eventos, se actualiza el modelo. 3. ejecuta el tick, es decir, se le hace saber a todos los objectos del juego que paso tiempo. 4. Obtiene los nuevos estados de los objectos del modelo y se los envía a cada cliente.
4. `GameControler` es la clase que relaciona los eventos enviados por el cliente con el modelo. Para cada tipo de evento ejecuta distintos métodos del modelo. Ademas, almacena los objectos relacionados al juego en si.
5. `Model` es un paquete que contiene a todas las clases que representan al modelo: `GameObject`, `WalkingUnit`, `Map`, `Building`, etc...

Para la comunicación entre el hilo del juego y el hilo del cliente se usa una Shaque (shared queue). El cliente pushea los eventos a la shaque y Game los popea en el paso 1 de la descripción de Game.

## Cliente

Las clases principales del Cliente son:

1. `Client` es la clase principal que se instancia con los parámetros recibidos por el usuario, conteniendo el host y el puerto correspondiente al Server. Esta clase realiza la conexión por el Socket y espera a recibir una comunicación por parte del Server indicándole que comienza el juego, junto con alguna configuración más (player_id y mapa del juego). Luego de ello, instancia tres nuevos threads: EventsLooperThread, EventsReceptorThread y EventsSenderThread. Además instancia los atributos compartidos por los distintos threads y los pasa por referencia. Estos se detallaran mas adelante.

2. `EventsReceptorThread` es un thread cuya única responsabilidad es recibir eventos del Server a través del Socket. Cómo la recepción de eventos es una función bloqueante, se decidió mover esta funcionalidad a un thread separado. Una vez que recibe un evento, este es agregado a una SharedQueue de Eventos. Este thread terminará cuando se cierre la conexión con el Socket.

3. `EventsLooperThread` es el thread principal del juego y como su nombre lo indica realiza el Event Loop correspondiente a los eventos producidos por la intervención del usuario (click con el mouse, con flechas, etc). Este thread instancia un WindowController, que es la clase principal que controla la ventana, y realiza el Event Loop del juego, enviando los eventos que sucedan al WindowController. Pushea cada uno de los mensajes que se deben enviar al server a una BlockingQueue, de forma que la acción propia del envío de los mensajes es delegada. Además popea los updates recibidos por el EventsReceptorThread y utiliza esa información para actualizar la vista. Este thread terminará cuando se cierre la conexión con el Socket.

4. `EventsSenderThread` es un thread cuya única responsabilidad es enviar eventos del Cliente al Server a través del socket. Si bien el envío de mensajes no es una función bloqueante, se decidió mover esta funcionalidad a un thread aparte y utilizar una BlockingQueue, de forma que el thread no se quede en un ciclo while hasta que llegue un evento. Cómo este thread se queda bloqueado al esperar un nuevo evento en la BlockingQueue, se forzará la terminación del mismo desde la clase Client agregando un nuevo elemento a la cola (de forma que se desbloquee y falle al tratar de enviar el evento a través del Socket)

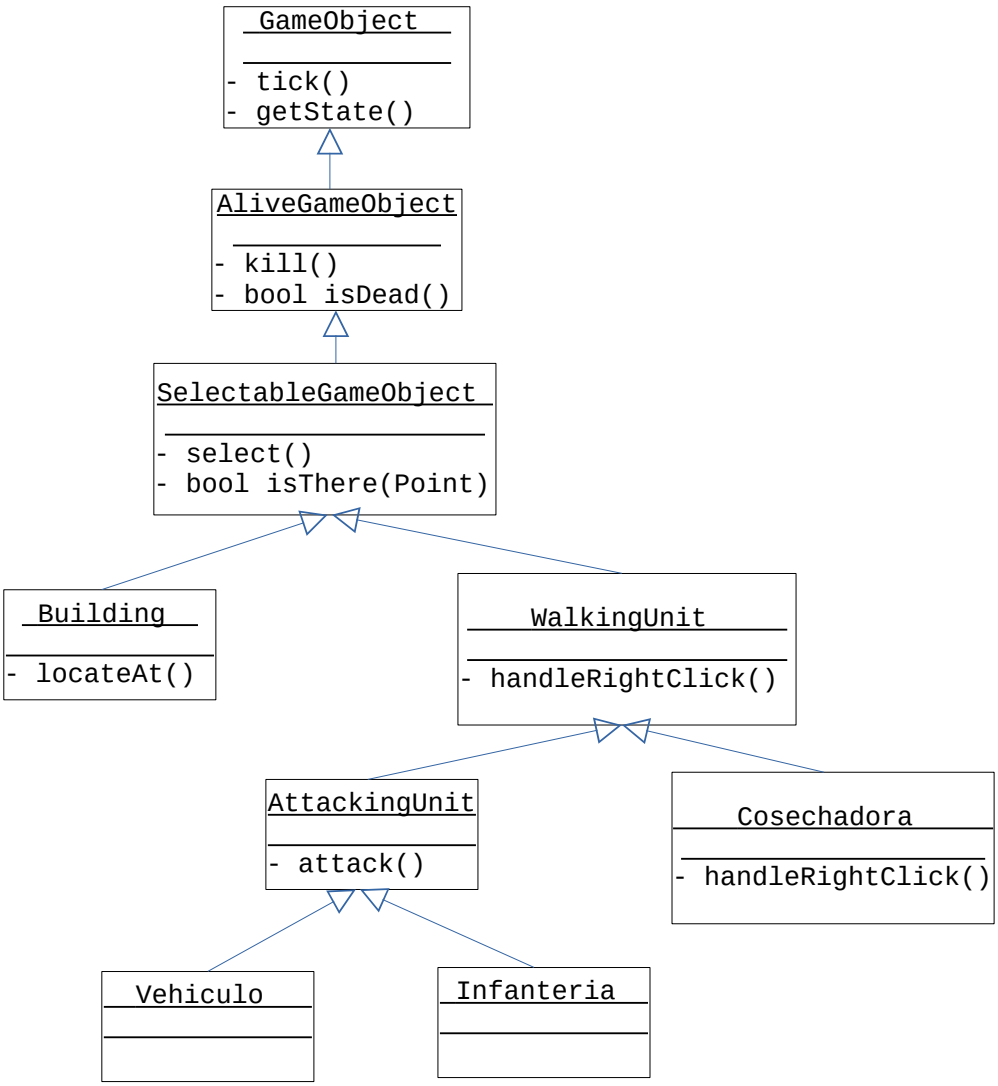
5. `WindowController` es la clase que controla todo lo que sucede en la ventana. Tiene asimismo otros 3 Controllers entre sus atributos: StatusController, TerrainController y ButtonsController. Cada uno de ellos se encarga de distintas secciones de la interfaz gráfica: la barra de estado conteniendo la energía y la especia, el terreno y el ojo de águila con todos los personajes y edificios, el panel de botones conteniendo todos aquellos elementos que pueden ser construidos por el jugador.

6. `StatusController` es la clase que controla todo lo que sucede en la barra de estado. Recibe el evento del WindowController y se encarga de actualizar y re-renderizar la energía y la especia de texto a imagen, en caso que esta se haya modificado.
7. `TerrainController` es la clase que controla todo lo que sucede en el terreno. Recibe el evento del WindowController y se encarga de actualizar y re-renderizar tanto el terreno como los personajes, los edificios, las acciones del usuario, etc. Además tiene la lógica para renderizar el ojo de águila sobre el panel de botones.
8. `ButtonsController` es la clase que controla todo lo que sucede en el panel de botones. Recibe el evento del WindowController y se encarga de actualizar y re-renderizar los botones disponibles para el jugador, así como también el progreso de construcción del item asociado al botón

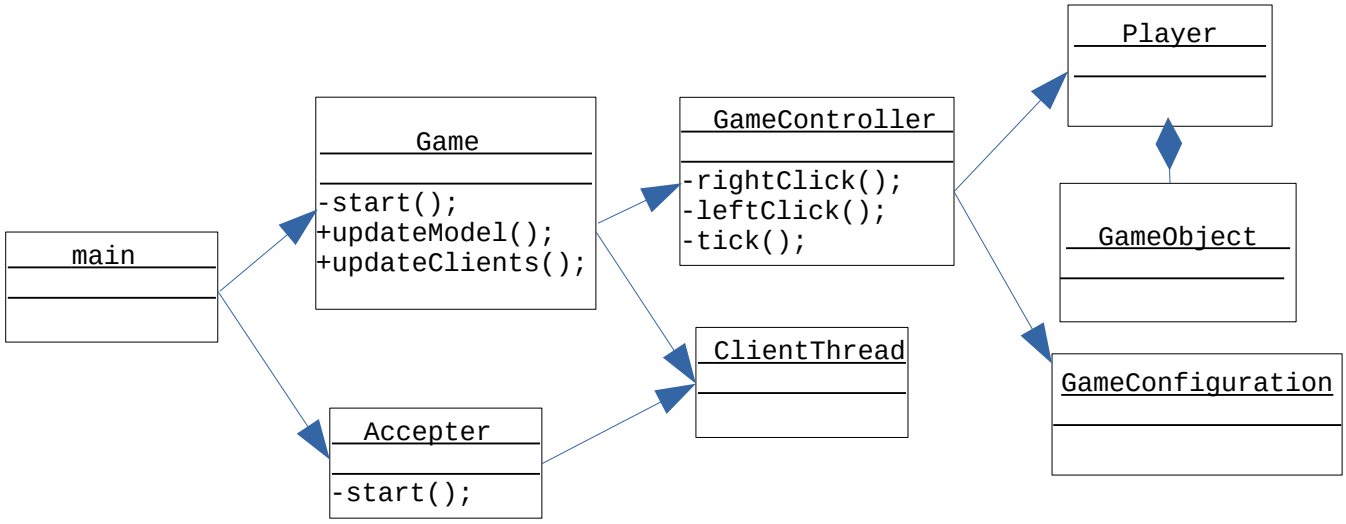
# Diagramas

## Diagrama 1: relaciones entre los objectos del juego.

Todos los objectos del modelo del juego están incluidos en esta cadena de herencia, las subclases van agregando funcionalidad a los componentes básicos.

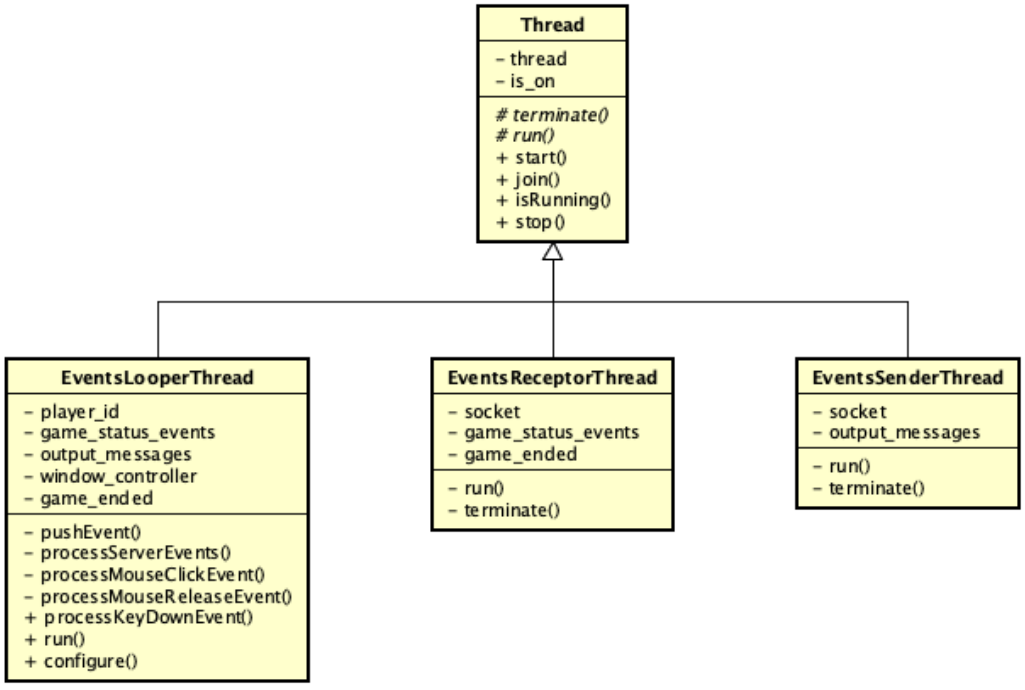


## Diagrama 2: componentes principales de la arquitectura del servidor



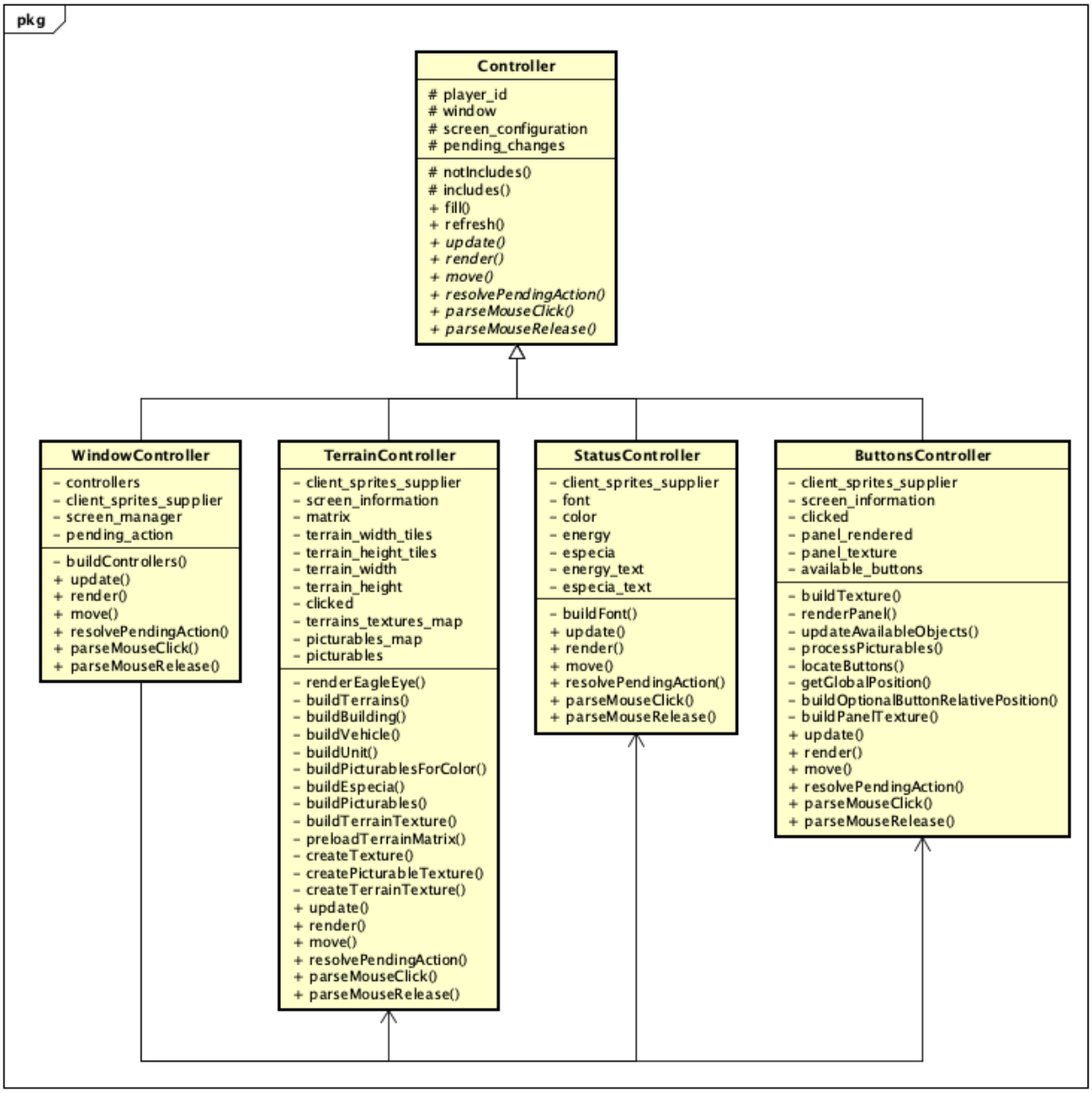

## Diagrama 3: Jerarquía de Threads

El Client se encuentra conformado principalmente por tres threads que corren en paralelo, con distintas responsabilidades:



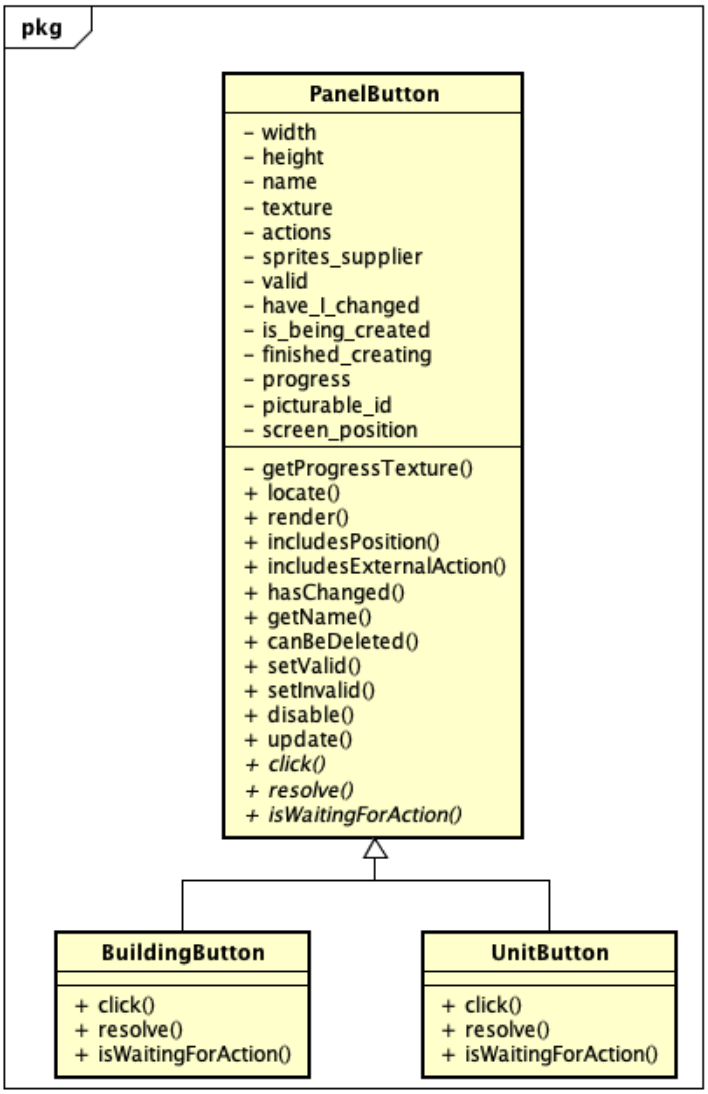
## Diagrama 4: jerarquía de Controllers en el Cliente

Los Controllers están diseñado de forma polimórfica. Cuando WindowController recibe un evento, este itera cada uno de los subcontrollers pasando el mensaje a cada uno. Es la responsabilidad de cada uno saber si tiene que actuar o no frente a cada evento. Además, cada controller recibe una función útil para pushear un evento directamente en la cola de eventos de salida. El único detalle aquí es que para la creación de edificios, se necesita una acción en el panel (click) y otra en el terreno (otro click). De hacer esta validación se encarga el WindowController.



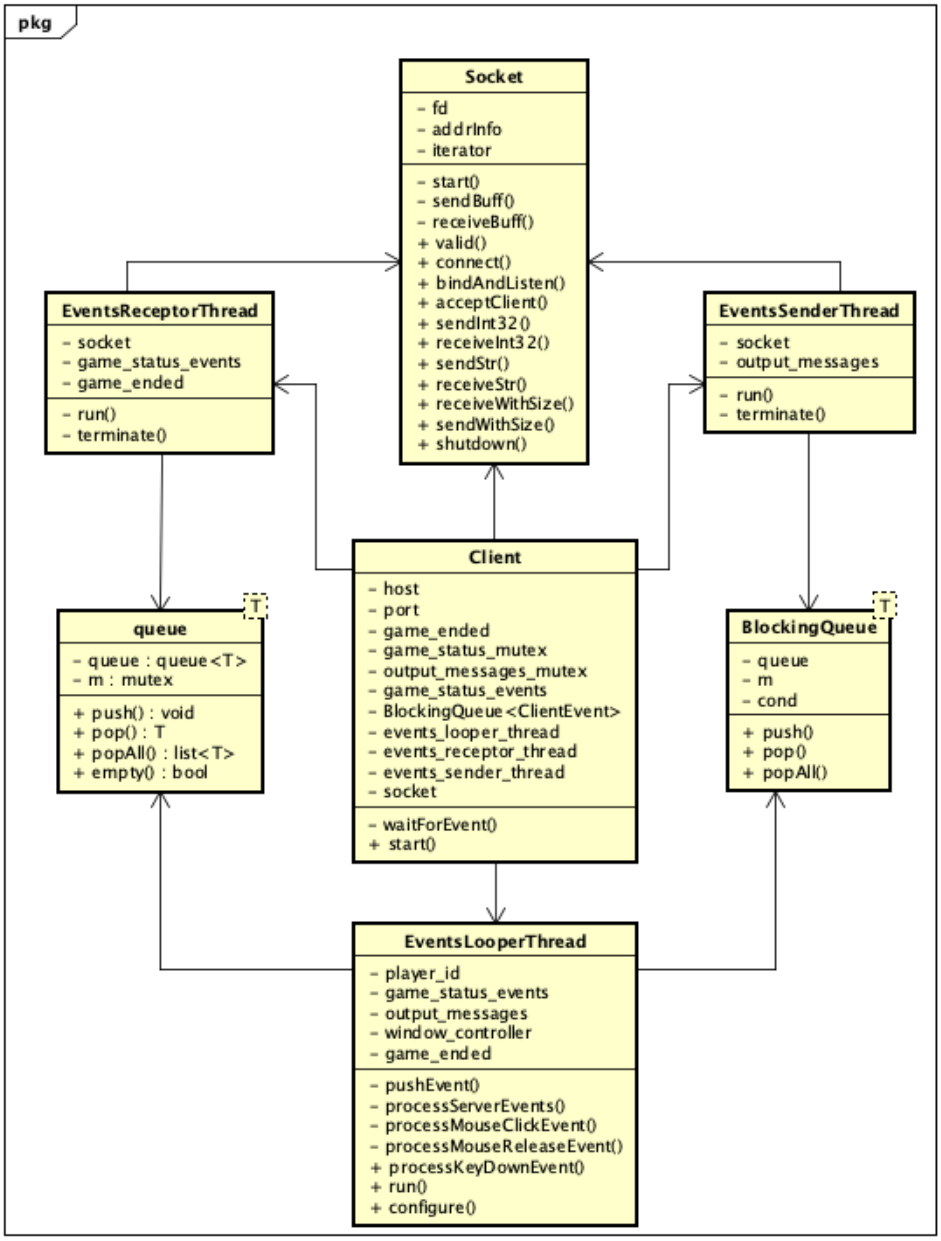
## Diagrama 5: jerarquía de Botones en el Cliente

Los Botones del Panel de la derecha han sido diseñados como Widgets, de forma que existen dos implementaciones: una para los edificios,que necesitan localizarse en un lugar específico, y otra para las unidades, que se localizan automáticamente.



## Diagrama 6: componentes principales de la arquitectura del cliente.

Los 3 Threads mencionados anteriormente tienen distintas responsabilidades. Corren en paralelo y se comunican a través de recursos compartidos y protegidos para evitar Race Conditions.



# Manual del Usuario

## Instalación

1. Descomprimir el Dune.zip.
2. Parado en la carpeta `cmake-build/` ejecutar `cmake..` y luego `sudo make install`.
3. Para ejecutar el servidor, desde la carpeta raíz del proyecto ejecutar `server server.conf`.
4. Para ejecutar el cliente, ejecutar `client <host> <puerto>`. Donde host es el la direccion donde se ejecuto el servidor y puerto esta detallado en el archivo server.conf. (8080 por defecto)

   Para poder jugar, primero se debe ejecutar un servidor. Luego los jugadores se conectan a este. Cuando la cantidad de jugadores conectados llega a la cantidad especificada en server.conf, comienza el juego.

## Juego

Cada jugador comienza con un centro de construcción y algunas unidades. El centro de construcción es el edificio mas importante, si este es destruido, se pierde el juego. Un jugador gana cuando todos los otros jugadores pierden.

Para pelear contra los enemigos, es necesario construir unidades de ataque que pueden atacar unidades y edificios de otros jugadores.

Cada jugador comienza con 3 de estas unidades, llamadas Trike. Para molerlas primero hay que seleccionarlas haciendo click izquierdo sobre ellas o arrastrando el mouse sobre ellas (presionando click izquierdo) para seleccionar varias a la vez. Luego hacer click derecho en la posición a donde se desea mover. Para atacar a una unidad enemiga, el procedimiento es el mismo pero en vez de enviar a la unidad seleccionada a una posición, se la enviá a una unidad o edificio enemigo.

Ademas, un jugador comienza con 3 cosechadoras. Estas sirven para recolectar especia y llevarla a los silos. La especia sirve para crear nuevas unidades. Para indicar a una cosechadora que recolecte especia, se debe seleccionar, y luego enviar a un bloque de especia. Las cosechadoras recolectaran una cantidad determinada de especia y luego la llevaran automáticamente al silo mas cercano.

La cantidad de especia que un jugador puede tener aumenta con la cantidad de silos y refinerías que posea.

Para crear nuevos edificios, se necesita energía. Estas son proporcionadas por las trampas de aire. Para crear trampas de aire, se debe seleccionar el icono correspondiente en el panel de la derecha. Así iniciara la creación del edificio. Una vez terminado, se debe seleccionar nuevamente el icono completado y luego en el mapa para especificar la ubicación del nuevo edificio. Los edificios solo se pueden colocar sobre roca.

Otros edificios se crean de la misma manera, pero se debe poseer suficiente especia y energía para crearlos.

Ademas de los Trikes, existen otros tipos de unidades. Para poder crearlas se debe crear antes el edificio de creación. Por ejemplo, para crear infantería ligera, es necesario poseer un cuartel.

Las unidades se crean igual que los edificios, excepto que son ubicadas automáticamente cerca de su edificio de construcción.

Un jugador puede vender sus unidades u edificios seleccionándolos y presionando la teca suprimir. Se le devolverá la mitad de su costo. Cuidado! Si vende su centro de construcción, perderá el juego.