Taller de Programación I - 75.42
Micromachines

Manual de Usuario

Índice

1. Requerimientos de hardware recomendados

- Procesador: AMD Athlon II x2 260, Intel Core2 Duo E8600 o similar.
- RAM: 2GB

2. Librerías necesarias

- SDL2
- Qt5
- ffmpeg
- Lua 5.3.4

2.1. Instalación

Ubuntu

Ejecute el siguiente comando en una terminal para instalar las librerías necesarias (a excepción de Lua)

```
$ sudo apt install build-essential libreadline-dev libsdl2-dev libsdl2-image-dev libsdl2-
    mixer-dev libsdl2-ttf-dev qtbase5-dev qtdeclarative5-dev qtmultimedia5-dev
    libqt5multimedia5-plugins libavutil-dev libswresample-dev libavformat-dev libavcodec
    -dev libswscale-dev libtiff-dev
```

Para instalar Lua ubíquese en una terminal en la carpeta donde desee descargar el instalador y ejecute los siguientes comandos:

```
$ mkdir lua_build
$ cd lua_build
$ curl -R -O http://www.lua.org/ftp/lua-5.3.4.tar.gz
$ tar -zxf lua-5.3.4.tar.gz
$ cd lua-5.3.4
$ make linux test
$ sudo make install
```

3. Instalación del juego

En una terminal, diríjase al directorio raíz del proyecto y ejecute la secuencia de comandos:

```
$ mkdir build
$ cd build
$ cmake ..
$ make
```

4. Uso

4.1. Ejecución

En una terminal en la carpeta `build` (ver instalación) ejecute el siguiente comando para ejecutar el cliente y/o servidor.

```
$ ./micromachines-client
$ ./micromachines-server
```

4.2. Uso del cliente

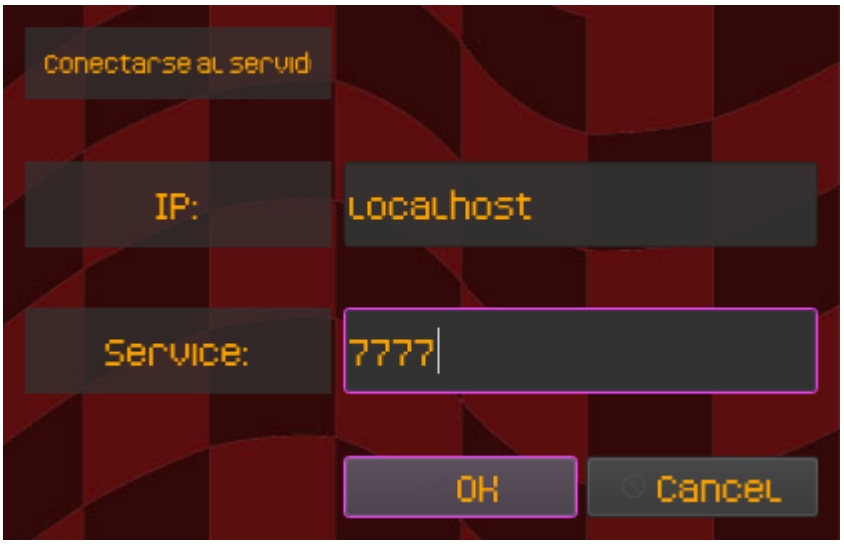

Figura 1: Pantalla de conexión: Deberá introducir el hostname (o IP) y el servicio (o puerto) del servidor al que desee conectarse

Figura 2: Menú principal del juego: Aquí debe seleccionar una opción, ya sea para crear una partida o unirse a una ya existente

Figura 3: Pantalla de creación de partida: Aquí deberá introducir el apodo que desee mostrar durante el juego, y el nombre de la partida que desee crear. También seleccionar el mapa que desee usar en su partida

Figura 4: Pantalla de selección de partida: Al igual que en la pantalla de creación de partida, deberá introducir su apodo, y seleccionar la partida a la que desee unirse, si se ha creado una partida recientemente, presione Actualizar Partidas para recargar la lista

Figura 5: Pantalla de espera: Si has creado una partida, cuando todos los jugadores se hayan unido presion ***empezar***, una vez que la partida haya iniciado no se podrán unir nuevos jugadores

Figura 6: Pantalla de espera: Si te has unido a la partida de alguien más, en esta pantalla esperarás a que el anfitrión inicie la partida.

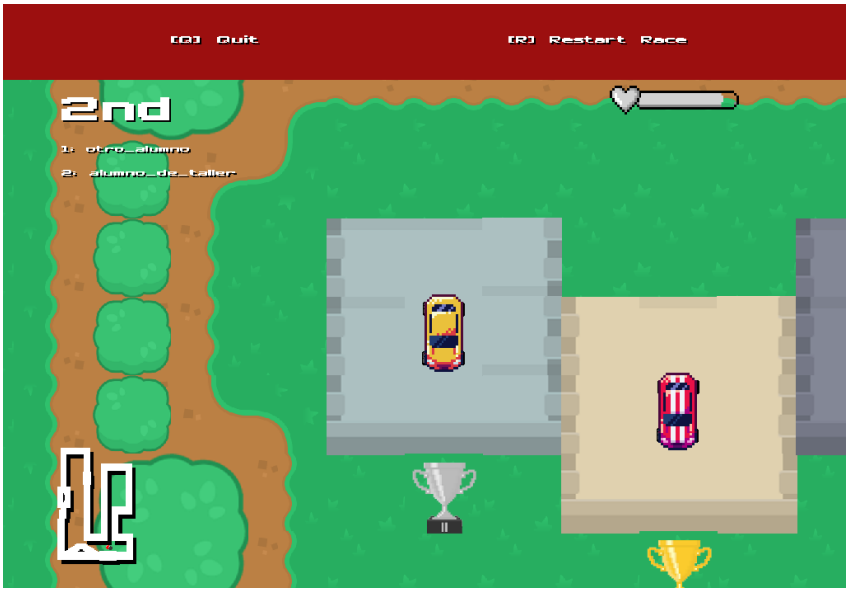

Figura 7: Si todos los jugadores conectados terminaron la carrera, el anfitrión verá un menú para reiniciar la carrera o terminar el juego. Si el anfitrión se ha desconectado en algún momento de la carrera se seleccionará otro anfitrión al azar.

4.3. Controles del Juego

- **Flecha Arriba**: Acelerar
- **Flecha Abajo**: Frenar/Marcha atrás
- **Flecha izquierda/derecha**: Virar el vehículo ó si ya has terminado la carrera, observar a otros jugadores
- **ESC**: Pausa/Menú de opciones
- **F9**: Iniciar/Detener grabación de partida
- **F11**: Pasar a modo pantalla completa/ventana

También está la posibilidad de controlar el vehiculo utilizando un joystick:

- **Boton 2**: Acelerar
- **Boton 3**: Frenar/Marcha atrás
- **POV/Hat**: Virar hacia la izquierda o derecha

Nota: Analógico no soportado.

4.4. Configuración del servidor

La configuración del servidor se encuentra en el archivo `configs.json` a continuación se encuentra una explicación de cada parámetro

- **max_players**: Máximo de jugadores por partida (no debe superar la cantidad de puntos de aparición o *spawnpoints* del mapa). La cantidad máxima permitida es de 8 jugadores. Cualquier valor por encima de 8 no se asegura el correcto funcionamiento del servidor.
- **server_port**: Puerto donde desee escuchar clientes.
- **map_path**: Ruta a la carpeta donde se encuentren los mapas relativo al ejecutable.
- **total_maps**: Cantidad de mapas existentes. Debe coincidir con el vector *maps*, caso contrario no se asegura un correcto funcionamiento del servidor.

- **maps**: vector con los mapas que se desean cargar. Debe coincidir el tamaño con la configuración *total_maps*, caso contrario no se asegura un correcto funcionamiento del servidor.
- **plugins_path**: Ruta a la carpeta de plugins relativo al ejecutable.
- **time_to_start**: Segundos del contador inicial de la carrera.
- **server_frames**: Cantidad de actualizaciones por segundo del servidor.
- **socket_pending_connections**: Tamaño de la cola de escucha del servidor.
- **max_len_name**: Máxima cantidad de caracteres para los nombres de los jugadores.
- **modifier_spawn_prob**: Probabilidad de aparición de un modificador en cada iteración del modelo. Suceden 60 iteraciones por segundo. (Valor por defecto: 0.005)
- **mod_oil_prob**: Probabilidad de que el modificador creado sea *oil*.
- **mod_mud_prob**: Probabilidad de que el modificador creado sea *mud*..
- **mod_fix_prob**: Probabilidad de que el modificador creado sea *fix*.
- **mod_rock_prob**: Probabilidad de que el modificador creado sea *rock*.
- **mod_boost_prob**: Probabilidad de que el modificador creado sea *boost*.
- **respawn_time**: Tiempo de reaparición de los jugadores luego estallar su auto (Unidad: segundos)
- **max_life**: Vida máxima del vehículo. (Valor por defecto: 100)
- **max_forward_speed**: Máxima velocidad hacia adelante del vehiculo. Puede no tener efecto si los atributos *back_wheel_max_force* y *front_wheel_max_force* no permiten alcanzar esa velocidad. (Unidad: m/s, Valor por defecto: 120 m/s)
- **max_backward_speed**: Máxima velocidad de reversa del vehiculo. Puede no tener efecto si los atributos *back_wheel_max_force* y *front_wheel_max_force* no permiten alcanzar esa velocidad. El valor debe ser negativo. (Unidad: m/s, Valor por defecto: -20 m/s)
- **back_wheel_max_force**: Fuerza máxima capaz de ser aplicada por las ruedas traseras. (Unidad: N, Valor por defecto: 120)
- **front_wheel_max_force**: Fuerza máxima capaz de ser aplicada por las ruedas delanteras. (Unidad: N, Valor por defecto: 150)
- **back_max_lateral_impulse**: Capacidad de cambio de dirección de las ruedas traseras. Cuanto menor sea el impulso, menos agarre tendrán las ruedas. (Unidad: N/s, Valor por defecto: 8.5)
- **front_max_lateral_impulse**: Capacidad de cambio de dirección de las ruedas delanteras. Cuanto menor sea el impulso, menos agarre tendrán las ruedas. (Unidad: N/s, Valor por defecto: 7.5)
- **use_dynamic_objects**: Habilita los objetos dinámicos en la pista. Esta opcion puede afectar el rendimiento dependiendo las especificaciones de la computadora que aloje el servidor o de la velocidad de internet.

4.5. Cerrado del servidor

Si durante la ejecucion del servidor se desea cerrar el servidor de forma segura, sin importar la cantidad de partidas corriendo o en espera, es posible realizarlo presionando la letra **q** seguido de **Enter**.

4.6. Docker

Se implemento un Dockerfile para crear una imagen modificada a partir de la imagen gcc de Docker. En esta imagen correra el server

En la raiz del proyecto ejecutar:

```
docker build -t micromachines-server .

docker run -p 7777:7777 -it --rm --name my-micromachines-server micromachines-server
```

5. Bot

El usuario tiene la posibilidad de utilizar un bot para que juegue por el. El scrip que hace posible el manejo automático del auto se encuentra implementado en Lua, sin embargo es posible que el usuario cree su propio script encargado de manejar el auto. Para que este script sea usado, es necesario que el mismo sea guardado en la carpeta lua donde se encuentra el ejecutable del juego, con el nombre bot.lua.

5.1. Bot API

Para que el script sea utilizado es necesario que cumpla con cierta interfaz que detallaremos a continuacion

- **init(keyDef, actionDef):** Esta función sera llamada una unica vez al inicializar la clase que controla al bot y recibira las teclas que puede accionar el bot y las acciones que puede realizar con esas teclas.
- **addTile(xi, xf, yi, yf, value):** Se llamara a esta función cada vez que el cliente agregue un tile al mapa. El parámetro value representa de que tipo sera el tile, ej: pasto, tierra, asfalto, etc.
- **updateCar(newCar):** Esta función es llamada cuando el cliente reciba la actualización del auto desde el servidor. Dentro de esta tabla podremos encontra posX, posY, rot, vel.
- **decide():** Por ultimo esta función se encarga de realizar una decisión que modificara o no el comportamiento del auto. Debe devolver una acción y una tecla de las recibidas en la funcion init. Es muy importante que se respete este orden.

6. Creación de mapas

Para la creación de mapas para este juego, se debe utilizar el software ***Tiled***, el cual puede descargar de `https://www.mapeditor.org/`. También se debe utilizar el *tileset* provisto.

Un mapa esta compuesto por 3 *layers* (capas).

6.1. Primera capa

En la primer capa se debe realizar el dibujo del mapa, Para un correcto funcionamiento de la lógica de la carrera, esta capa debe contar con:

- **Spawnpoints o puntos de aparición**

Este *tile* se puede utilizar en cualquiera de sus cuatro rotaciones y va a marcar el punto de inicio de un auto en la carrera. Se debe colocar uno por cada vehículo que correrá la carrera. Este tile se puede encontrar en el *tileset* con el ID 8.

- **Podio - Primer puesto**

Este *tile* posee solo la orientación hacia abajo. Se puede encontrar en el *tileset* con el ID 131.

- **Podio - Segundo puesto**

Este *tile* posee solo la orientación hacia abajo. Se puede encontrar en el *tileset* con el ID 136.

- **Podio - Tercer puesto**

Este *tile* posee solo la orientación hacia abajo. Se puede encontrar en el *tileset* con el ID 141.

- **Podio - Puesto no clasificado**

Este *tile* posee las cuatro orientaciones, pero se recomienda utilizar la posición hacia abajo, ya que se podrá visualizar mejor el auto en la pantalla final. Se puede encontrar en el *tileset* con el ID 36.

- **Linea de llegada**

Este *tile* posee las cuatro orientaciones. Para generar correctamente la linea, se debe colocar ambos bordes. La orientación de conteo de vueltas sera indicado en la segunda capa.

- **Tile Inicio de pista**

Este tile se utiliza en el algoritmo que determina las posiciones. Sin el mismo, no es posible determinar las posiciones de los vehículos. Debe agregarse justo después de la linea de llegada, preferentemente en el centro.

En esta capa también se pueden agregar los objetos estáticos, como tribunas, arboles, lagos, etc. La información de si un objeto es estático se encuentra en las propiedades del *tile* en el *tileset*

6.2. Segunda capa

En esta capa agregaremos una propiedad a cada tile. A los tiles que conforman la pista se les podrá asignar una orientación (que permitirá restablecer el auto en la posición correcta o en el caso de la linea de llegada, determinar la orientación para el conteo de vueltas). Para los tiles que conforman el entorno se les podrá asignar la propiedad de **limite**. Los tiles con la propiedad de limite explotan al vehículo, evitando que pueda salir del mapa. A todos los demás tiles del entorno se debe asignar la propiedad **NoLimite**.

- **Arriba (UP)**

- **Abajo (DOWN)**

- **Derecha (RIGHT)**

- **Izquierda (LEFT)**

- **Tile limite**

- **Tile no limite**

6.3. Tercer Capa

En esta capa se introducen los objetos dinámicos, es decir aquellos objetos con los cuales podemos interactuar durante la carrera. En la versión actual estan implementados apenas tres objetos:

- **Ruedas rojas**

- **Rueda blanca**

- **Cono**

7. Plugins del servidor

A la hora de generar un plugin se contara con cierta información brindad por el server en forma de DTO y se deberá respetar cierta interfaz para que la ejecución pueda ser correcta.

7.1. Compilación

A la hora de compilar un plugin se deberá realizar utilizando los flags necesarios para generar una librería dinámica la cual no dependa de la posición del programa en memoria.

7.2. Agregar Plugin

Para que el juego ejecute el plugin creado por el usuario, es necesario que el mismo se encuentre en la carpeta Plugins, la cual es creada al momento de compilar el juego. Una vez que la librería compilada se agregue a esta carpeta y el server sea reiniciado, automáticamente se incluirá entre los plugins actuales.

7.3. Plugin API

Para el correcto funcionamiento y ejecución de un plugin es necesario que el mismo cuente con tres funciones las cuales se detallaran a continuación

- **void* init()** Esta función se encarga de inicializar los atributos necesarios para el plugin. Estos atributos serán guardados en memoria dinámica y esta función se encarga de devolver un puntero a la posición de memoria donde se encuentren estos atributos.
- **void execute(void*, DTO_Info*)** Para realizar la ejecución de nuestro plugin se llamara a la función execute la cual recibirá un void* que apunta a los atributos generados en la función init y un puntero a la estructura DTO_Info la cual se detallara mas adelante
- **void destroy(void*)** Por ultimo, nuestro plugin deberá tener una función destroy que se encargara de liberar la memoria pedida por la función init

Es de extrema necesidad que estas funciones cumplan con las firmas de las funciones especificadas arriba por lo que en caso de ser el pulugin programado en C++ es necesario que se use extern Ç" para declarar dentro las funciones mencionadas.

7.4. DTO_Info

DTO_info es el tipo de estructura de dato que recibirá nuestro plugin para poder saber la información de la carrera actual y para poder modificarlo y asi poder alterar la carrera. Las componentes de esta estructura se detallaran a continuacion:

- **cars:** Nos dice la cantidad total de autos que se encuentran corriendo la carrera
- **total_laps:** La cantidad de vueltas que componen una carrera
- **car_info:** Estructura que contiene la información de cada uno de los autos
  - **ID:** ID del auto al cual pertenece la información dentro de esta estructura
  - **laps:** Cantidad de vueltas recorridas por este auto
  - **specs:** Aquí se encuentra la información que usa el servidor al momento de crear el auto
    - **max_life:** Se detalla en la sección configuraciones

    - **max_forward_speed:** Se detalla en la sección configuraciones
    - **max_backward_speed:** Se detalla en la sección configuraciones
    - **backK_wheel_max_force:** Se detalla en la sección configuraciones
    - **front_wheel_max_force:** Se detalla en la sección configuraciones
    - **back_max_lateral_impulse:** Se detalla en la sección configuraciones
    - **front_max_lateral_impulse:** Se detalla en la sección configuraciones
- **life:** Esta estructura contiene la información de la vida del auto
    - **max_life:** Tiene la vida máxima que puede tener el auto
    - **current_life:** Aquí se encuentra la vida actual restante del auto
- **position:** Posición actual del auto con respecto al resto de los autos.
- **modifier:** Aquí se encuentra el modificador a aplicar