## Introduction:

This document describes a resource manager component that can be used in any game with ease. It contains a set of examples for using this component, the features of the component and a description of how to demonstrate the component.

## How to use this component:

### Example: Create and Initialise the resource manager

// Include the resource manager

#include “ResourceManager.h”

// Declare the resource manager and get an instance

ResourceManager\* m\_resourceManager = ResourceManager::getInstance();

// Initialise the resource manager with the renderer

m\_rm->init(SDL\_Renderer\* m\_renderer);

### Example: Load from file

// Load all assets passing the name of the text file that contains the

// location of the assets

m\_resourceManager->loadResourcesFromText(“fileName.txt”);

m\_resourceManager->loadResourceQueue();

Example of the format the Text document must use:

texture // The type of resource

player\_texture // The key used for the texture

Assets/Textures/player.png // The path to the texture

texture

enemy\_texture

Assets/Textures/enemy.png

music

game\_music

Assets/Music/game\_music.ogg

music

end\_music

Assets/Music/end\_music.wav

sound\_effect

jump

Assets/SoundEffects/jump.wav

sound\_effect

land

Assets/SoundEffects/land.wav

animation

stick\_man

Assets/Textures/sprite\_sheet.png

4 // The number of frames in the animation

64 205 0 0 // The frame in the format width, height, x, y

64 205 64 0

64 205 128 0

64 205 196 0

// Load all assets passing the name of the JSON file that contains the

// location of the assets

m\_resourceManager->loadResourcesFromJSON(“fileName.json”);

m\_resourceManager->loadResourceQueue();

Example of the format the JSON document must use:

{ "resources": {

"textures":

{

"player":

{

"key": "player\_texture",

"path": "Assets/Textures/player.png"

},

"enemy":

{

"key": "enemy\_texture",

"path": "Assets/Textures/enemy.png"

}

},

"music":

{

"game":

{

"key": "game\_music",

"path": "Assets/Music/game\_music.ogg"

},

"end":

{

"key": "end\_music",

"path": "Assets/Music/end\_music.wav"

}

},

"effects":

{

"jump":

{

"key": "jump",

"path": "Assets/SoundEffects/jump.wav"

},

"land":

{

"key": "land",

"path": "Assets/SoundEffects/land.wav"

}

},

"animations":

{

"stick\_man":

{

"key": "stick\_man",

"path": "Assets/Textures/sprite\_sheet.png",

"frames": 4,

"metaData":

{

"frame1":

{

"width": 64,

"height": 205,

"x": 0,

"y": 0

},

"frame2":

{

"width": 64,

"height": 205,

"x": 64,

"y": 0

},

"frame3":

{

"width": 64,

"height": 205,

"x": 128,

"y": 0

},

"frame4":

{

"width": 64,

"height": 205,

"x": 196,

"y": 0

}

}

}

}

}}

// Load all assets passing the name of the XML document that contains the

// locations of the assets

m\_resourceManager->loadResourcesFromXML(“fileName.xml”);

m\_resourceManager->loadResourceQueue();

Example of the format the XML document must use:

<?xml version="1.0" encoding="UTF-8" ?>

<resources>

<textures>

<texture>

<key>player\_texture</key>

<path>Assets/Textures/player.png</path>

</texture>

<texture>

<key>enemy\_texture</key>

<path>Assets/Textures/enemy.png</path>

</texture>

</textures>

<audio>

<music>

<key>game\_music</key>

<path>Assets/Music/game\_music.ogg</path>

</music>

<music>

<key>end\_music</key>

<path>Assets/Music/end\_music.wav</path>

</music>

</audio>

<sound\_effects>

<effect>

<key>jump</key>

<path>Assets/SoundEffects/jump.wav</path>

</effect>

<effect>

<key>land</key>

<path>Assets/SoundEffects/land.wav</path>

</effect>

</sound\_effects>

<animations>

<animation>

<key>stick\_man</key>

<path>Assets/Textures/sprite\_sheet.png</path>

<metaData>

<frame>

<width>64</width>

<height>205</height>

<x>0</x>

<y>0</y>

</frame>

<frame>

<width>64</width>

<height>205</height>

<x>64</x>

<y>0</y>

</frame>

<frame>

<width>64</width>

<height>205</height>

<x>128</x>

<y>0</y>

</frame>

<frame>

<width>64</width>

<height>205</height>

<x>196</x>

<y>0</y>

</frame>

</metaData>

</animation>

</animations>

</resources>

### Example: Get texture and render it

// Add this to get a texture from the resource manager

SDL\_Texture\* \_playerTex= m\_resourceManager->getTextureByKey("player\_texture");

// Clear screen

SDL\_RenderClear( renderer );

// Get the width and height of the texture

int \_w, \_h;

SDL\_QueryTexture(\_playerTex, NULL, NULL, &\_w, &\_h);

// Create a source rectangle for the texture

SDL\_Rect \_src = new SDL\_Rect();

\_src.h = \_h;

\_src.w = \_w;

\_src.x = 0;

\_src.y = 0;

// Create a destination rectangle for the texture

SDL\_Rect \_dest = new SDL\_Rect();

\_dest.h = \_h;

\_dest.w = \_w;

\_dest.x = 600;

\_dest.y = 600;

// Render the texture

SDL\_RenderCopy(m\_renderer, \_playerTex, &\_src, &\_dest);

// Update screen

SDL\_RenderPresent( renderer );

### Example: Get music file and play it

// Add this to get a music file from the resource manager

Mix\_Music\* \_gameMusic = m\_resourceManager->getMusicByKey("game\_music");

// Add this to play the audio asset

if (Mix\_PlayingMusic() == 0)

{

// Play the music

if (Mix\_PlayMusic(\_gameMusic, -1) == -1)

{

std::cout << "Problem playing game music!!" << std::endl;

}

}

### Example: Get sound effect and play it

// Add this to get a sound effect from the resource manager

Mix\_Chunk\* \_jump = m\_resourceManager->getSoundEffectByKey("jump");

// Add this to play the sound effect

if (Mix\_PlayChannel(-1, \_jump, 0) == -1)

{

std::cout << "Problem playing jump sound effect!!" << std::endl;

}

### Example: Get animation and render it

// Add this to get the sprite sheet and the frames from the resource manager

pair<SDL\_Texture\*, vector<SDL\_Rect>> \_stickMan =

m\_resourceManager->getAnimationByKey("stick\_man");

// Add this to get the current frame for the animation

SDL\_Rect \_src = \_stickMan.second[m\_currentFrame % \_stickMan.second.size()];

// Create a destination rectangle for the frame

SDL\_Rect \_dest = SDL\_Rect();

\_dest.w = \_src.w;

\_dest.h = \_src.h;

\_dest.x = 300;

\_dest.y = 300;

// Render the current frame

SDL\_RenderCopy(m\_renderer, \_stickMan.first, &\_src, &\_dest);

// Change the frame every quarter of a second

if (m\_animationDelay >= 0.25f)

{

m\_currentFrame++;

if (m\_currentFrame >= \_stickMan.second.size())

m\_currentFrame = 0;

m\_animationDelay = 0;

}

### Example: Reload Assets

// In order to reload assets you must call the update function of the resource // manager, passing the delta time as an argument. Every three seconds, this

// will check each of the textures and reload it if the file has been changed. // It also reloads the xml, json or text file used if the have been changed.

m\_resourceManager->update(\_deltaTime);

## Features:

### Load From file

* Loads from text file
* Loads from JSON file
* Loads from XML file

### Reload assets without restart

* Monitor underlying file changes
* Any change to an asset should cause the application to reload that asset

### Map assets

* Use a key-value map to store assets
* Use name of asset as key

## Demonstration of this component:

* Pressing 1 loads from a text file
* Pressing 2 loads from an xml file
* Pressing 3 loads from a json file
* Pressing ‘d’ will delete the resource manager and re-create it to test other load functions
* Pressing ‘p’ will play/pause the game music
* Pressing ‘j’ will play the jump sound effect
* Pressing ‘l’ will play the land sound effect
* Open texture in paint.net, change something and save it to see the changes in the application