

Sdl runner Design Document

Advanced Digital Game Programming



Lit Tipperary

Student: Derek O’Brien

Lecturer: James Daly

# Contents

## Introduction page2

## Back Story page2

## Key Challenges page2

## Technical Design Decisions page3

## Features Implemented page3

## Class Diagram page5

Design Doc for SDL Endless Runner

# INTRODUCTION

The game in which I plan to make is a 2d side scrolling endless runner where the player will have to jump and slide to avoid objects which if they collide with will kill them. My main focus of this game is to learn the SDL framework and implement a Data Driven Design pattern to read all sprite, background and menu icons from and XML file while building a free runner game. I also plan to implement the Data Driven Design pattern to save the players score as a value key pairing and display a high score table on the end game scene.

# Back Story

Backstory is simple you are just a player trying to avoid objects. My reason for this is I plan to concentrate more on the mechanics of the game and the game design its self. If I get time near the end of the project I will then elaborate on a backstory for the game.

# KEY CHALLENGES I FACED

Some of the main challenges I have faced throughout the duration on this project are:

* The Dao was one of the challenges I faced as I had no previous experience in implementing a design to load game data in this way. To overcome this challenge I broke down the task and completed it one section at a time and over time I got a better understanding of how the whole process worked and was able to implement it to suit my needs.

# Technical Design Decisions

For the technical side of the game play the player I have implemented three main mechanics which will are:

* Jump to jump over obstacles and collect collectables.
* Slide to slide under the obstacles.
* Super-Size to gain immunity for a short time.

I think that jump and slide are the two core player mechanics for any endless runner. The player will also have a power up mechanic which will be unique to my game. The unique player power up mechanic I have implemented is super-size. When the player activates the power up by picking up the shield the player will grow to double its size and be able to smash through one enemy. The disadvantage of this power up is the player loses ability to jump and slide when the power up is active.

The other unique mechanic I have implemented in my game is character selection before the game begins. To implement this mechanic I use the xml file and Dao. When the user selects the player they want to use this is saved to the xml file and on start of game this user decision is read from the file and corresponding sprite images are loaded.

For loading each of the game assets I use the Dao I created to load the sprite path from an xml file. To load the asset details from the xml file I designed the Dao in such a way that only the assets name needs to be passed into the function, the Dao read function will then locate the xml element that matches the name passed in and return its details. This procedure is also used to load and display all menu & hud buttons along with the sound assets.

Using the abilities of xml I have also implanted a mini value key pair database where I save and read the players current and high score from. The value pair key database saves the value passed into the function with the unique key as a pair in the xml file. This unique key is then used when I want to get the value back form the xml file.

I have also implemented what I think is a good core game structure with all game elements inheriting from a base game object. At the first stage of inheritance I split up my game objects in three main types:

1. Label: This label class is the base for all in game labels.
2. Layer: This layer class is the base for all game layers, I then future refine it to two new class which are HudLayer and Scrolling Layer.
3. Sprite: This sprite class is the base for all game elements, then future refined to button, collectables and characters. The character class is then refined to my two types of characters player and non-controllable player (NPC).

There are four scenes in the game a splash screen which loads up the game window and displays the game name and then leads to the main menu scene. On the main menu scene the user has the option to select which character they would like to play the game with by clicking on the player icon, this also starts the game scene. The users choice is then saved to the xml file and corresponding sprite loaded when game scene is entered. In the game scene there is an In-Game Hud where the player can pause the game or exit the game and also view its score and number of coins collected within the current game. Once the player dies the player’s current game score is saved to the xml file and it is checked against the last high score to see if it is a new high score, if it is the current score is then saved as the new high score. The game then proceeds to an end game scene where the game score is displayed along with the high score, the player will be presented choices to return to main menu to restart or else they can exit the game.

As I wanted to give the impressing of depth to the game scene I have implemented three different scrolling layers all moving at different speeds. This gives the impressing of a parallax background and the impressing on some depth as the game play exists in between two layers.

My initial design proposal was to use tile map for the background but as I had no previous knowledge of reading in an Xml file this proved quite difficult to me, so I reverted back to having a png image background.

# Features I Got Implemented

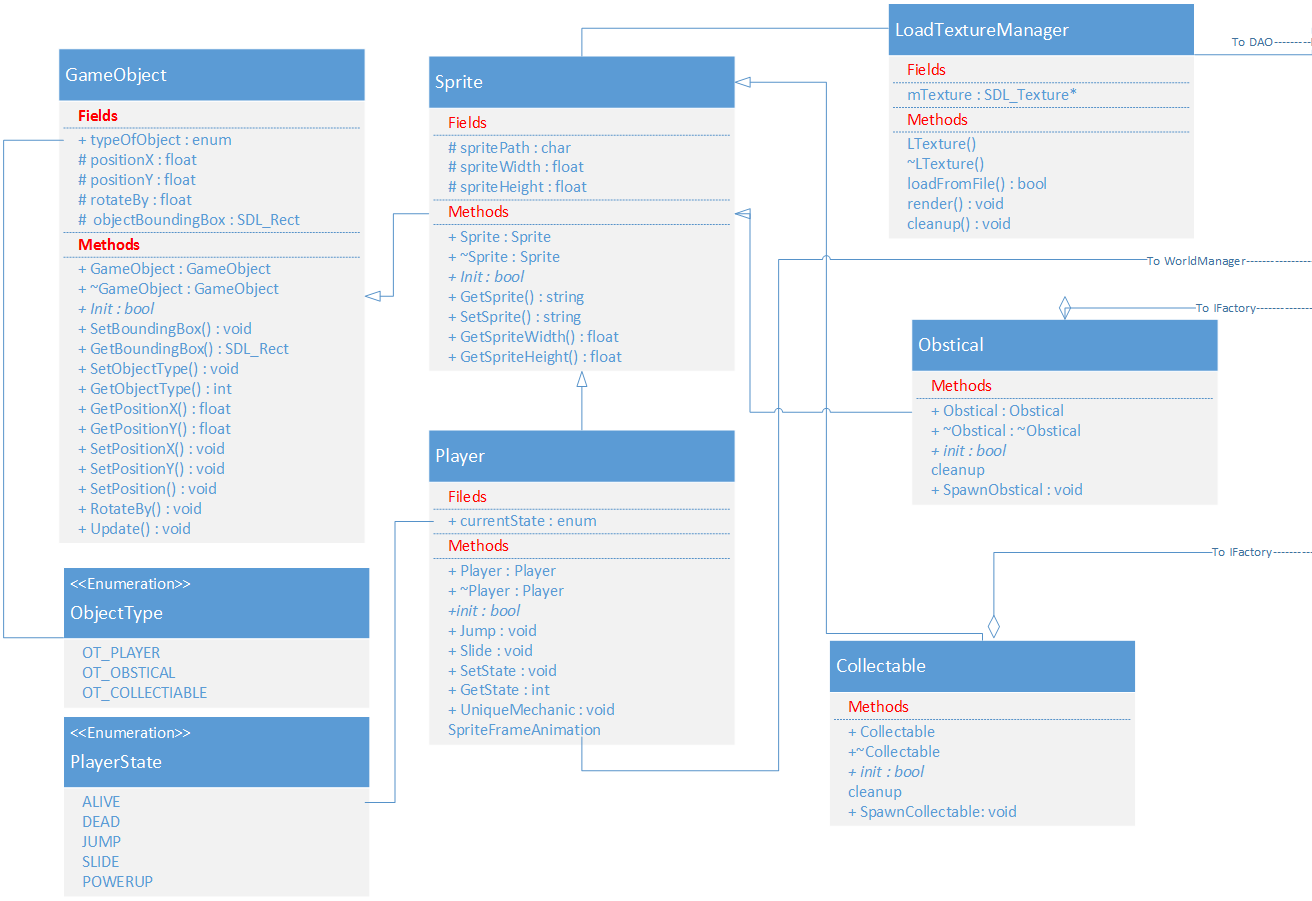
From the rubricks for this assignment I got the following features implemented:

* Scrolling background
* Collision & State Management (Alive, Dead etc.)
* Animation (Sprite frame coming from xml file)
* Data driven design (Some form of UI element being designated from a file)
* Scene Switching (Main menu to in-Game)
* Human interface controls
* In-Game HUD (With clickable pause functionality)
* 1 Unique Game mechanic (No two people in the class can use the same mechanic this could range from: rocket boosters, teleportation, invincibility etc.)

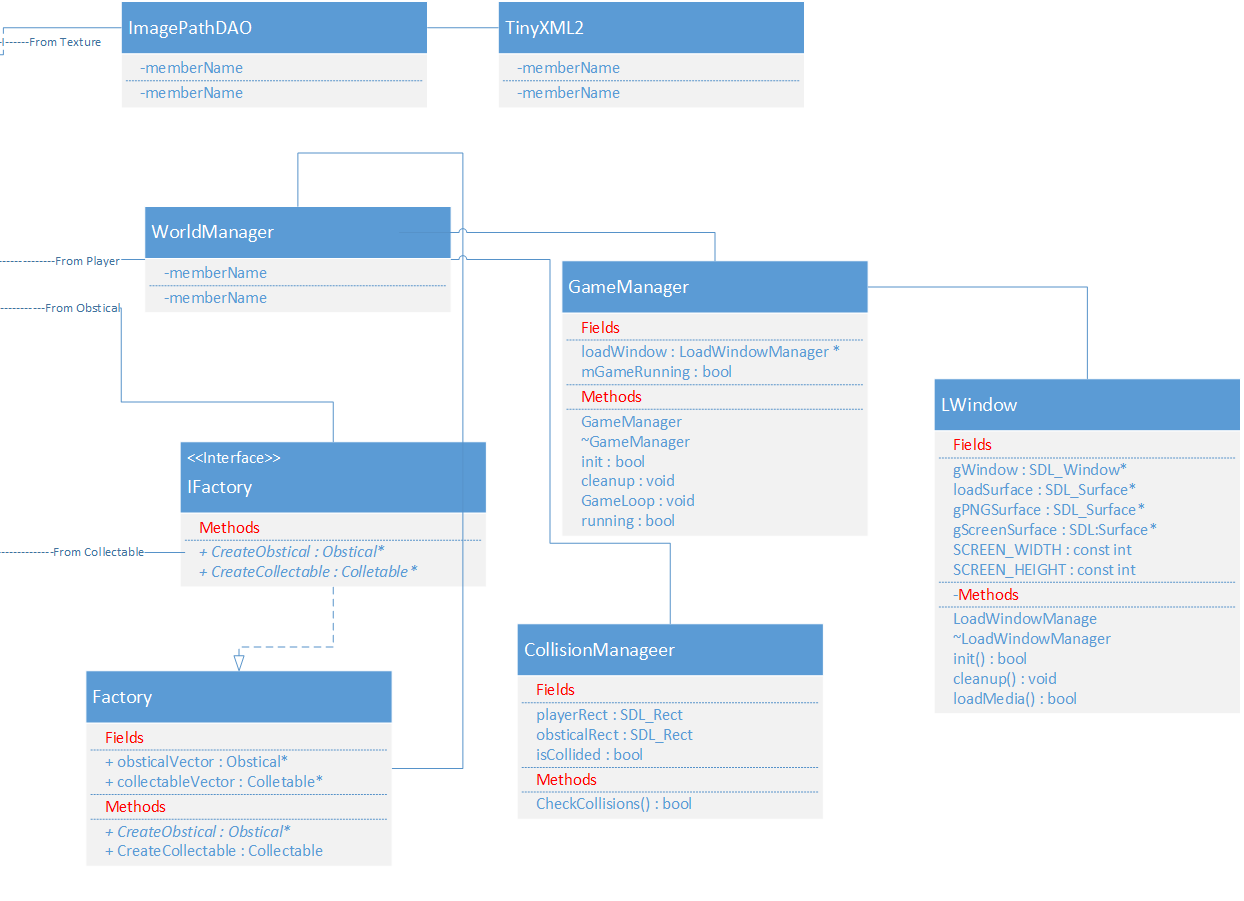
# CLASS DIAGRAM

In this document I have include my initial planned class diagram from before development started and I have include the completed project class diagram to show the difference from my planned structure to final structure. Fig 5.1 shows first half of initial class diagram, Fig 5.2 shows second half of initial class diagram.

Finished game class diagram can be found externally in the same folder as this document as it was too large to fit in the document



**Figure 5. 1 - Initial Class Diagram (Before development started)**



**Fig 5.2 – Second half on initial class diagram (Before development started)**