International Journal of Interdisciplinary Innovative Research & Development (IJIIRD)

ISSN: 2456-236X

Vol. 03 Special Issue | 2019

Light Weight 2D Game Engine for Action Games

Prof.Vijayalakshmi Kadroli[1], Ms.Anuja Desai[2], Prof. Safia Sadruddin[3], Ms. Ashwini Misal[4], Ms. Sanika Parab[5], Ms.Swapnali Bhisade[6]

[1, 2, 3, 4, 5, 6] Department of Information Technology, TPCT'S TEC, Nerul, Mumbai, India

ABSTRACT

Game Engine provides some very basic functionality enough to create simple 2D games, on a bare metal system. The creation of computer games requires a lot of knowledge and experience. Nowadays there exists a lot of frameworks that allows developers to create the games in faster and easiest way. This work presents the approach of creating screen scrolling 2D game using existing platforms (box2d, sdl). Game engines are called as the middleware as they help us to get business sense of the term also they help us to build a flexible and reusable software platform which helps us to provide all the core functionality needed by the user. It helps us to have an out of the box approach to develop a game application also it helps us to reduce the cost of game development and complexities are reduced.

1. INTRODUCTION

Nowadays, game is one of the entertainment media that has become a trend. Currently, game is not only used as a medium of entertainment, even some are made for the purpose of learning media with high effectiveness.

Two-dimensional games were most frequently developed in the early years of video games with the main reason for this being that the technical limitations of game hardware prevented the ease of creating three-dimensional graphics. When technology developed sufficiently to allow easier and more effective use of 3D graphics there was a temporary decline in 2D gaming. Many researchers have discussed the possible benefits for using games in education and we believe they may be particularly helpful in improving computer science education. Games provide a way to create and share educational content while also making students feel their computing education is more relevant. The purpose of these notes is to present and build in a logical progression the skills required to build a 2D game and specifically a platform game in C++. Game engine usually provides us the platform for game abstraction i.e. to run the same game on different platforms. Game engines are designed in such a way that they help us to replace specific systems with more specialized middleware components

2. GAME CONTROL MODEL

The game control model presented by authors was developed based on their experience and studies and represents a game ontology from an interactive content viewpoint. They mainly were focused on role-playing and simulation game genres, but the design can also be used in game genres such as action, adventure, sports, vehicle, strategy, etc. The game control model consists from ten interrelated key concepts that best represent the information about game: *Game Structure, Game Presentation, Game Rules, Game Scenario, And Game Object.*

2.1 Game Engine Structure

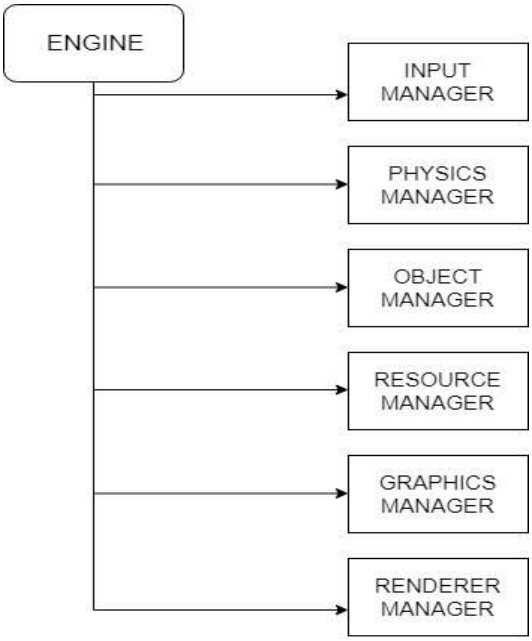

Fig 2.1 Game Engine Structure

The game engine structure is divided into 6 main components which are as follows
1.Input Manager
2.Physics Manager
3.Object Manager
4.Resource Manager
5.Graphics Manager
6.Renderer Manager

2.1.1 Input Manager

It is the data that the user provides to the system at the run time using any input device. The structure of the input device and the data provided by the device changes from one device to another. The work of this manager is to read the input device data and then extract the essential data required and pass this data in the game logic. The input manager also helps us by providing an abstraction layer to access this input device in a system. This layer calls the functions present in the hardware dependent input library of the specific device to get the input data

- **Input Map**

Every game should contain of an input map that defines which input events should correspond to what actions are to be performed. The table below shows some input actions and their respective keys

INPUT	***ACTIONS***
Key 3 is pressed	Start the game
Key 2 is pressed	Move to left
Key 4 is pressed	Move to right

2.1.2 Physics Manager

This manager is used to give the realistic behavior of the object in the game so that the objects in the game must accelerate correctly and should also be affected by external forces like gravity and friction. The physics manager performs two main tasks.

- Simulate movement

It helps to simulate the movement of objects. For example, walking, running or jumping. It also helps us to simulate the external forces like gravity, friction

- Collision detection

It helps us to detect intersection of two or more objects that are present in the game played by the user.

The scope of the manager is very vast.

2.1.3 Object Manager

The object manager is the smallest part of the game that interacts with the user. The logic of all the games is written around these objects and thus the object manager performs different tasks on these objects.

2.1.4 Resource Manager

The game engine needs to deal with resources like images and sounds to design the game. When we make use of the images and sounds then it enhances the gaming experience and thus makes it more real and making it memorable. Thus, it helps the user to pay attention to the game. To store the high-resolution sound and images we require more memory. And thus, we store this resources in flash memory. However, this flash memory is of very limited size and thus we are able to store only limited file. The resource manager will help us to store the resources needed by the game developer

2.1.5 Graphics Manager

It provides a interface which is helpful for creating a menu window. Graphics manager is divided into modules
1. Game Graphics
2. Graphical User Interface

2.1.6 Renderer Manager

The rendering process is the final process of displaying graphics on the LCD screen. It helps us to solve the complex mathematical calculations which help us to create life like graphics. And this is done by adding textual, lightning and shading information.
Some of the features of renderer manager is

1. **Shading**: helps us to know how the color and brightness of surface varies with lighting
2. **Reflection:** helps us to generate mirror like or highly glossy reflection.'
3. **Transparency or Opacity:** this feature helps us to know the transmission of light through solid objects
4. **Shadows:** the renderer manager will interact with the LCD controller to render graphics on the LCD on screen

2.2 Game Engine Presentation

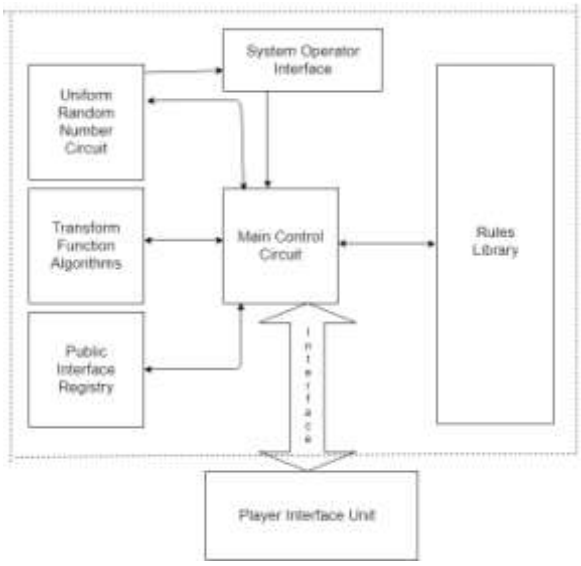

Fig 2.2 Game Engine Presentation

The simplified format of the gaming apparatus is shown in the above diagram. This apparatus is been created to match the present invention of the games which is also called as universal gaming engine.Game engine are used to perform several basic functions. One of the most basic function of the system is to detect the players initials event and then communicate every state of the game to the player. The game engine must always process the player initated event and thus determine the appropriate rules.The use of Main Control Unit is to control and perform the basic functions. This Main Control Unit is a software or hardware controlled

programmable microprocessor or microcontroller. This Main Control Unit can also be implemented as an ASIC device with dedicated logics present in it to perform the required control functions. The Main Control Unit is also used to communicate with the player interface unit via the interface bus.

The player interface unit acts like a machine that helps us to form a display for communicating the given information to the player.The player interface is then generating a player record of information and transmits this player record over the bus so that it reaches to the Main Control Unit. The Main Control Unit also responds to thr user initiated event. This initiation is done by rederencing with the Public Interface Registry. The Public Interface Registry is also importatnt because it is used as a lookup table which is implemented in volatile, semi volatile or non volatile memory. The Public Interface Registry is also said to be an addressable memory where each address is also associated with the mapping record. Thus the Main Control Unit uses this mapping record to address the Rules Library.

The Rules Library is used to address memory preferably to allow random access. The Main Control Circuit then processes the selected rules by selectively accessing the Random Circuit and then Transform the Function Algorithms. The completely deterministic rules are executed totally within the Main Control CIrcuit and this is done by the calculations or data transfer operations. Wherein this rules require Main Control CIrcuit access to one or more Pseudo-Random Numbers, Random Number Circuit is accessed. The Preferred Embodiment Random Number Circuit is used to provide a series of Pseudo Random Numbers of arbitary length having uniform distibution.A rule will always require a non uniform distribution of Pseudo-Random Numbers or can also need a subset of the Numbers. In some cases the Main Control Unit will implement the rule that is selected by accessing Transfer Function Algorithm from the block. The System Operator Interface is used by the game developer which is used for the communication with the Uniform Random Number Circuit and Main Control Circuit and the Public Interface Registry. The system operator interface is used to enable the operator which is used to initialize, monitor and thus change the seed values and the key values which are been used by the Uniform Number Circuit. We can make use of any hardware to implement System Operator Interface including a DIP switche, terminal , and a PC. Now to implement the game the programmer will develop a series of rules that are required for the game to function properly.

This rules are then stored in volume in Rules Library. The programmer will then be registering the new game which is in the Public Interface Registry for storing the location of the volume of the rules in a proper address in Public Interface Registry. The player is using the Player Interface Unit and thus can access any of the games stored in the Rules Library. Whenever we want to create a new game then the game regulation authority will only review the rule of the Rule Library and thus verify them weather they follow the given rules. The Player Interface Unit are totally electronic or combination of electronic and mechanical components. This interface will submit any amount of information with the basic function set.

3. GAME DEVELOPED ON THE GAME ENGINE

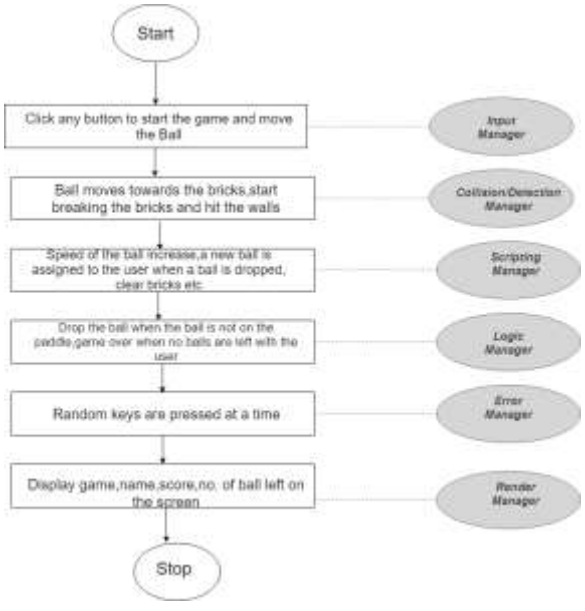

Fig 3.0: Game development Algorithm

The figure shown above is a example game that is been made using the game engine that is developed. The game developed from this engine is called as **ARKNIOD.** Each function of the game is linked with the managers of the game engine and perform specific actions. The diagram shown above is a generic moudle diagram for the arkniod game.

4. CONCLUSION

Hence, a game engine provides a platform for the implementation and development of 2d games, allowing the creators to use the easy-to-learn tools with the engines own scripting language. And thus we are able to develop a light weight game engine for 2d action games

5. REFERENCES

[1] "Secure OTA vehicle software update" IEEE paper published on 25 November 2015
[2] "Secure FOTA Object for IoT" in 2017 IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops).
[3] "INTERNET OF THINGS (IOT) SECURITY BEST PRACTICES" IEEE paper published on February 2017.
[4] "MQTT based home automation system using ESP8266" in 2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC).
[5] "Working Paper Updating firmware of embedded systems in the Internet of Things" 62nd meeting, 27-28 November 2017, Paris (France)
[6] Arm limited "A Firmware Update Architecture for Internet of Things Devices" https://tools.ietf.org/id/draft-moran-fud-architecture-00.html
[7] "An Over the Air Update Mechanism for ESP8266 Microcontrollers" in ICSNC 2017: The Twelfth International Conference on Systems and Networks Communications.
[8] Texas Instruments: "Implementation of IrDA with the MSP430" http://www.ti.com/lit/an/slaa202a/slaa202a
[9] NS8: Device fingerprinting description https://www.ns8.com/ns8u/what-is/device-fingerprinting