A Review on Developing an Arcade Game Machine and an Arcade Game using Raspberry Pi and Pygame

Nishant Sahni
Computer Engineering Department, Mukesh Patel School of Technology Management & Engineering, NMIMS University, Mumbai, India

Kailash Srinivasan
Computer Engineering Department, Mukesh Patel School of Technology Management & Engineering, NMIMS University, Mumbai, India

Harsh Savla
Computer Engineering Department, Mukesh Patel School of Technology Management & Engineering, NMIMS University, Mumbai, India

ABSTRACT
In this paper, we will discuss and review the steps involved in developing an arcade game machine from ground-up along with designing and developing an arcade game to run on it. We will also discuss the choice of hardware and the development tools used for developing our system. In-game physics will also be incorporated using the PyODE engine and Pygame libraries. We will also compare Easel and Pygame as game development libraries and will determine which would be the most appropriate for our project.

Keywords
Arcade Game, Raspberry Pi

1. INTRODUCTION
The technical details of Raspberry Pi and elucidation of how they will be used in the proposed system have been mentioned in this paper. Further, every aspect of the proposed system has been mentioned in detail including the OS installation and various commands for the configuration of the Raspberry Pi processor. The first part of the paper deals with the hardware, where all the problems associated with Raspberry Pi and its corresponding Raspbian OS installation are discussed. The second part of the paper deals with the software, where we discuss the development of the Arcade game which will be coded in Python using Pygame Libraries. The in-game physics would also be implemented using the PyODE engine that is supported by this library.

2. BRIEF DESCRIPTION
We will now discuss the development of the Arcade game machine and the Arcade game in two different sections:

2.1 Arcade Game Machine
This machine would be an embedded system whose sole purpose would be to run the arcade game we design for it. It's hardware components are described as follows:

2.1.1 Raspberry Pi:
Raspberry Pi would be the most appropriate processor for our system. It is small in size and has low power consumption. We will be using the B+ model which flaunts a clock speed of 700MHz in normal conditions and 1000MHz in Turbo. It has a 512MB SDRAM and a 128MB GPU [4]. These specifications are more than enough for us to be able to smoothly run any arcade game. Also, it has an in-built HDMI port which will enable us to connect it to a display device. Raspberry Pi also supports external input which would play a major role in our system. General purpose input/output are a set of generic pins on the processor whose behaviour can be controlled as well as programmed through software. It has a Micro SD card slot which will enable us to boot the OS as well as the game. It is also relatively cheap and thus reduces the overall development cost of the system.

2.1.2 Display Device
This would display the general purpose output of the system. Any standard display device which has an A/V or an HDMI port can be used for our system. It may range from an old CRT display to the newer LCD display.

2.1.3 Cabinet
This would be the external casing of our system so that it stays protected from external entities and looks more visually appealing. We will use Medium-density fibreboard for making this cabinet.

2.1.4 Heat Sinks
These are passive heat exchangers used for cooling the device by dissipating heat into the surrounding medium. We will be using a heat sink to keep our Raspberry Pi processor from overheating.

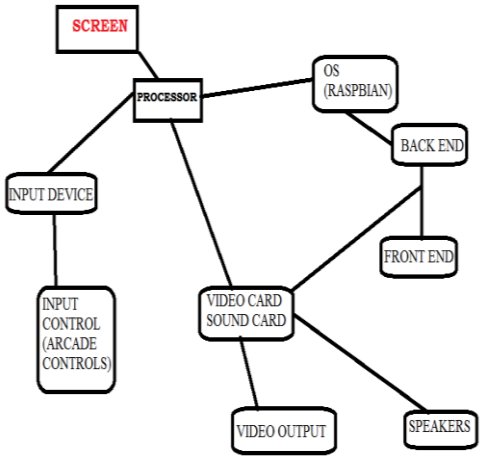
2.1.5 Controls
These would provide a medium for the users to interact with the system. They would be responsible for the general purpose input of our system. The controls would comprise of buttons and a joystick.

After the components are put together we must initialize the system with the appropriate software so that it is capable of running the designed arcade game. The steps involved for doing so are as follows:

1. We must download the latest RetroPi SD card image and write the image to the SD card so that we can boot it from our Raspberry Pi processor. RetroPi is a preconfigured setup for the Raspberry Pi running EmulationStation, which is basically a front end for navigating and launching games for multiple emulators [5]. We will need at least a 4GB card just to run RetroPie.
2. We must then load our game onto the SD card over the network. Game ROMs are stored in the appropriate folder on the RetroPi and will be automatically read by EmulationStation.

3. The last part would require us to configure the controls. The physical controls need to be wired before we can tackle this part. It involves identifying the ports that each button is wired to and then editing the *controls config* file to map the keys to the right buttons in-game.

The basic structure of our arcade game machine can be best illustrated with the help of the following block diagram:



We will now venture into the steps to be followed and tools required for developing the arcade game which our machine is designed to play.

2.2 Arcade Game

Developing the arcade game would be the most time consuming task. We will be using Python as the programming language to develop our game as it has a lot of freely available game development libraries. Also, it is a high-level language, which means better code readability and more efficient debugging. It is possibly the best language for writing game-world simulations in. It's clear to read and write, easy to learn, handles a lot of programming house-keeping and is reasonably fast [2]. Using Python would also enable us to use PyODE which is an engine for in-game physics. Two main game development libraries are described as follows:

2.2.1 Easel

It is a framework for creating real time games by defining pure functions [3]. It was designed principally for the purpose of game programming. Unfortunately, Easel does not support in-game physics and hence is not appropriate for our project.

2.2.2 Pygame

SDL is a multimedia library that allows access to hardware in a cross platform fashion. The PyGame library is a set of Python bindings to the SDL library. It consists of various top level "pieces" like Sprite, Surface, Font, Mixer etc. each of which is necessary to create a full game [2]. It provides user input handling through mouse, keyboard or joystick and game output via screen for shape drawing, font rendering, etc. and speakers for sound effects and music. Pygame strictly supports 2-D graphics. Since we are not dealing with 3-D graphics, this restriction of Pygame does not affect our project. Hence, with its controller support and in-game physics engine PyODE, it is easily the most appropriate game development library for our system.

Now that the game development library has been selected, let us look into developing physics aware games using these libraries.

2.2.3 PyODE

Here ODE stands for Open Dynamics Engine. It has mainly two features:

2.2.3.1 Rigid Body Simulations

Its first and main feature is Rigid Body Simulations. PyODE simulates the effects of various forces on various bodies. The specific shapes of the bodies are not particularly relevant. Physical properties like mass, mass distribution, etc. are important. To use this library, we first create a world and set its properties like gravity etc. We then create the bodies. The library provides us some useful calls to create objects of different shapes by specifying their parameters (eg. Creating a sphere by specifying its radius, mass and density) [1]. It must be noted that the shape is useful for the system to calculate the centre of gravity. Once it figures that out, it simply treats the object as a point mass and changes its position and orientation according to the effects of the various forces acting on it.

2.2.3.2 Collision Detection

This is PyODE's secondary feature. It helps in detecting when two objects collide, with what force did they collide and how the objects will react after the collision.

The two features of PyODE are independent of each other but work well in combination.

With the help of the above mentioned tools we will develop our arcade game. This game would then be transferred over the network to the SD card in our Arcade machine.

2.3 Advantages & Disadvantages

2.3.1 Advantages of Existing Techniques

- Made physics aware games easier to implement
- All of this was possible due to the proper use of the PyODE engine
- Rigid body simulations were properly implemented

2.3.2 Disadvantages of Existing Technique

- Physics simulation may or may not be accurate
- Event handling (mouse clicks), sounds etc. are difficult to manoeuvre
- One fault in the loop of the Pygame code will disturb the entire system
- Basically laws of physics are not very easy to be implemented in a Pygame.

2.3.3 Advantages of Proposed System

- As python is used, coding is easier
- Pygame library provides almost all facilities required for implementing an arcade game
- The components proposed for the system are readily available
- Use of heat sinks helps the processor to maintain its heating capacity

2.3.4 Disadvantages of Proposed System

- Providing heat sinks may make the system bulky
- Embedding the input controls (arcade controls) to the system is not an easy task
- Placing the RetroPi in a specific location is a tedious job

- Online gaming is not incorporated

3. INFERENCE

This paper barely scratched the surface of the universe that is independent gaming. A myriad of high quality libraries are there on the internet which can make your life as a game developer easy. There are multiple libraries to choose from when looking to create games. ODE is a full-fledged 3D physics engine. It might be too heavy for our needs and there are alternatives like chipmunk and box2d both of which are 2D engines. PyGame is implemented as a raw C extension to Python and totally relies on SDL. The concept of collision detection is well explained in the existing system. ODE can be made compatible to 2D engines by making minimal use of it in the system. Through ODE, various mathematical calculations can be used to resolve the laws of physics that are implemented in rigid body simulations

4. CONCLUSION

As far as system requirement is concerned, the Raspberry Pi processor provides all the basic functions provided by the desktop. The type of model for our system is the B+ model that is very versatile, providing high Debian friendly OS. As far as Easel and Pygame is concerned, no doubt that Easel is more user-friendly, but because of its main disadvantage of not supporting in-game physics, Pygame does it better. The game to be developed would support physics ODE engine through which developing the physics related rigid simulation system becomes easier, which is the main concern of our project.

5. REFERENCES

[1] Noufal Ibrahim KV, "Creating Physics Aware Games using PyGame and PyODE" in The Python Papers Monograph 2: 20, Proceedings of PyCon Asia-Pacific 2010.

[2] Richard Jones, "Rapid Game Development In Python".

[3] Josh Archer, Bryant Nelson, and Nelson Rushton, "An Experiment Comparing Easel with Pygame".

[4] Dhaval Chheda, Divyesh Darde, Shraddha Chitalia, "Smart Projectors using Remote Controlled Raspberry Pi" in International Journal of Computer Applications (0975 – 8887) Volume 82 – No 16, November 2013.

[5] Rolfebox, "2-Player Bartop Arcade Machine (Powered by Pi)", at www.instructables.com.