

DSP - Digital Simulated Pinball

David Hawkey (djh4227@rit.edu)
Ashlyn Zoecklein (adz3234@rit.edu)

October 29, 2007

Contents

1	Overview	3
2	Input Controls	4
2.1	Control Interface	4
2.2	Ball Shooter Interface	4
3	Operational Specs	7
3.1	Minimum Requirements	7
3.2	Power Requirements	7
3.3	System Maintenance	7
4	User Interface	8
4.1	Playfield	8
4.2	Backboard Display Screens	9
5	Music System	10
5.1	Overview	10
5.2	Custom Music Playlist	10
6	Analytical	11
6.1	Power Calculations	11
6.2	Physics System (Newton) Input Parameters	11
6.3	Flipper Physics	12
7	Software	13
7.1	Operating System	13
7.2	Software Overview	13
7.3	Interprocess Communication	14
8	Performance Measurements and Testing Strategies	16
8.1	Performance	16
8.2	Testing	16
8.2.1	Controls/Inputs	16
8.2.2	Shared Memory	17
8.2.3	Pinball Application	17
9	Project Expenses	17
9.1	Manufacturability	17
9.2	Item Costs	18
10	External Concerns	19
10.1	Safety Considerations	19
10.1.1	Epilepsy	19
10.1.2	Motion Sickness and Headaches	19
10.2	Environmental Considerations	19

11 Summary	19
12 Source Code	20

1 Overview

The digital simulated pinball project is a twist on classic pinball, using 3d computer graphics and flat screen monitors to simulate the traditionally mechanical playfield. The game resembles an actual pinball machine, including actual pinball controls (buttons, coin acceptor, and ball shooter), with a large flat screen television in place of the playfield, and a smaller flat screen on the backboard used to simulate a dot matrix display. A general purpose computer is used to run the hardware simulation which runs on Linux and uses several open source libraries for graphics and sound.

2 Input Controls

2.1 Control Interface

The pinball controls are interfaced to the computer using a low-cost USB gamepad device. The gamepad was an ideal choice because it is already designed for use in gaming applications and support is already built-in to the operating system so it requires no special drivers. The input can also be abstracted through the Object Oriented Input System (OIS).

The gamepad has many buttons built onto several circuit boards which are attached to a main circuit board by ribbon cables. The surface-mount buttons act as resistors that are high impedance when released and low impedance when depressed. The built in buttons were detached from the controller, and the new controls were soldered onto the control board.

2.2 Ball Shooter Interface

Interfacing the ball shooter proved to be a particular challenge since a sensor needed to be installed that could accurately detect the position of the shooter while not interfering with the feel and high-velocity release.

Initial tests were conducted using a linear potentiometer that had a travel distance equivalent to that of the ball shooter. The resistance range also matched that of the analog joystick on the gamepad interface device, so a direct connection could be made. This method proved to be troublesome for several reasons. There was a relatively low accuracy for determining position, and the results were not consistent. There was a significant dead-zone region where changes in position would not register at all. Fast movements would also frequently not be detected, which was unacceptable for a ball shooter release.

An improved ball shooter interface was designed that utilizes infrared transmitters and receivers to detect the position of the ball shooter. The infrared modules are positioned linearly along the ball shooter shaft such that moving the ball shooter will block or allow the infrared light from the transmitters to reach the receivers. A circuit was designed to convert the voltage from the IR receivers into acceptable voltage levels for the gamepad controller buttons. At the rest position, the ball shooter blocks all the receivers, resulting in all gamepad buttons being released. As the ball shooter is pulled back, connections are made between the IR modules resulting in gamepad button presses. There are currently 4 sensors, allowing for 4 discrete levels of position detection. It was determined that this was an appropriate number to allow for a realistic pinball shooter. There is also a small level of randomization added in software for the shooter so the ball will have a slightly different velocity on every shoot even if pulled back to the same exact level, therefore increasing the range of ball shoot velocities despite the position quantization of the hardware.

The infrared receivers needed to be interfaced with the gamepad controller so that an activated receiver causes a button press on the controller. This was accomplished by creating a small circuit that converts the infrared receiver output to digital logic levels 0V and 5V for off and on respectively. The NPN transistor is used as a switch, the current from the phototransistor into the collector causes a change in resistance across the collector and emitter. The changes in resistance will pull

the collector to ground, resulting in a low logic level. A schmitt trigger was added to clamp the output levels to the correct 0V and 5V levels. Due to available parts on-hand, a hex inverter with schmitt trigger was connected with another hex inverter, which has the same result as only a schmitt trigger. The value of the 6.76K resistor on the phototransistor was found by adjusting a potentiometer until an appropriate output response was obtained (correct logic levels for blocked and unblocked states).

The infrared LEDs are mounted directly across from the receivers, such that the ball shooter blocks the light from reaching the receiver when the shooter is in its rest position. Pulling the ball shooter back will unblock the receiver. The LEDs are connected to 5v with a small current limiting resistor.

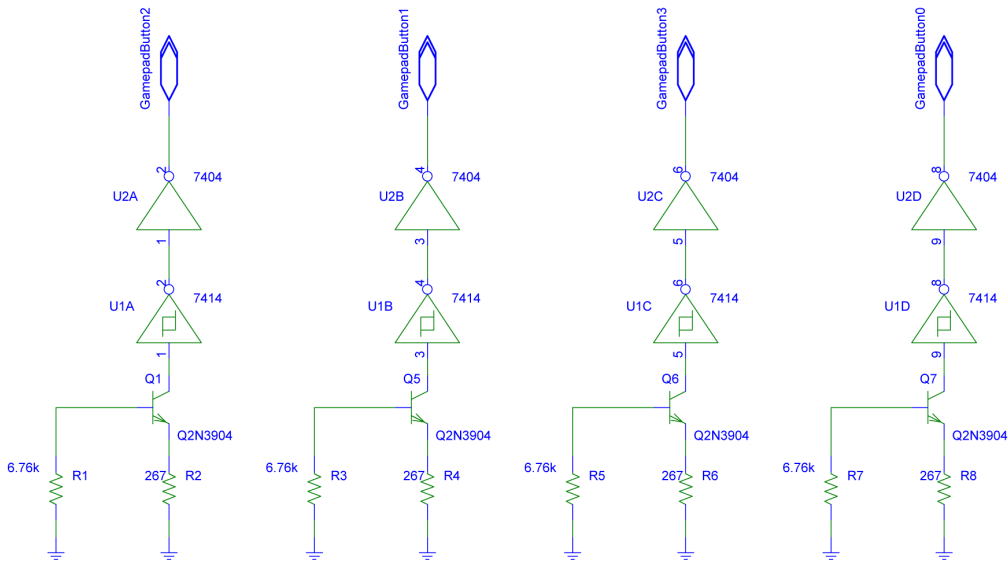


Figure 1: Infrared Receiver Interface Circuit

$$V_{CEOMAX} = 70V \quad (1)$$

$$V_{ECOMAX} = 5V \quad (2)$$

$$I_{CMAX} = 50mA \quad (3)$$

$$SpectralBandwidth = 620 - 980nm \quad (4)$$

Figure 2: Infrared Detector Specifications

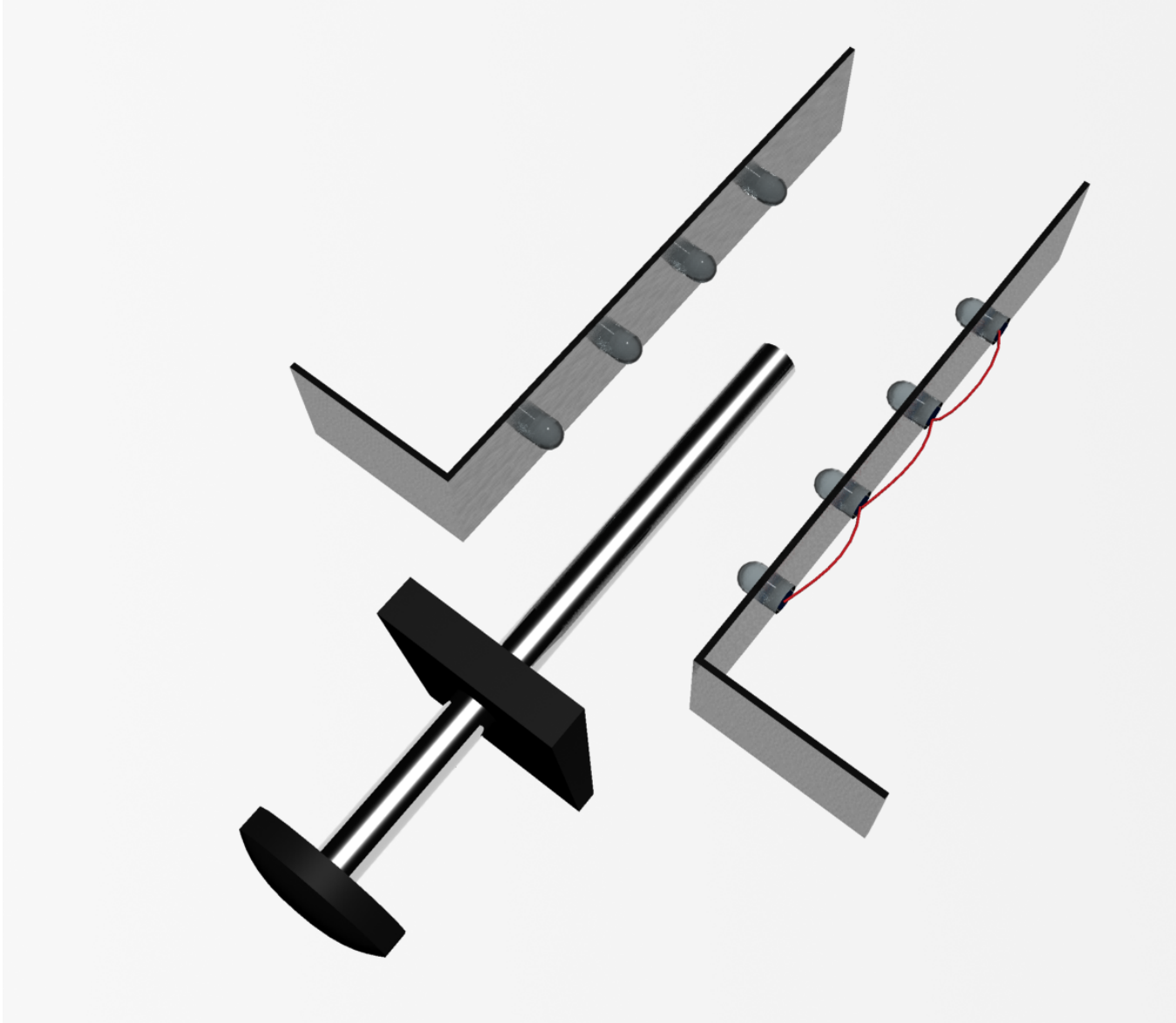


Figure 3: Ball Shooter Infrared Configuration

3 Operational Specs

3.1 Minimum Requirements

Pentium 4 2.26 Ghz or equivalent
2 Displays
NVIDIA 7600 Series
512mb of RAM
Ubuntu 7.04

3.2 Power Requirements

Component	Max Power Consumption
Processor (Pentium 4 2.26Ghz)	100W
Video Card (NVidia 7600)	70W
Memory (2 DDR)	20W
Sound Card	7W
Network Card	4W
USB Controller	5W
System Fans	4W
Enclosure Fan	4W
Total	214W

Figure 4: Power requirements of components used.

3.3 System Maintenance

There are several components of the pinball machine that have an estimated lifecycle and may require replacement after this time. Even though the components are rated for long-life, they may require periodic service especially in high-traffic environments where the machine will be used frequently (game rooms, casinos, etc.). These components are designed into the pinball machine so that they can easily be replaced. Figure 5, shown below, lists these components. All of these components are critical to the successful operation of the pinball game.

Component	Estimated Lifespan	Replacement Cost
Flipper Buttons	10,000,000 Cycles	\$5
Start Button	10,000,000 Cycles	\$6
Start Button Lamp	15,000 Hours	\$1
Coin Acceptor Mech	N/A	\$22
Ball Shooter	15,000 Cycles	\$2

Figure 5: Lifecycle and replacement cost for components used in the pinball machine.

4 User Interface

4.1 Playfield

The playfield has a music inspired theme. There are many elements that are designed to make the game more interactive and fun while allowing players to achieve high scores.

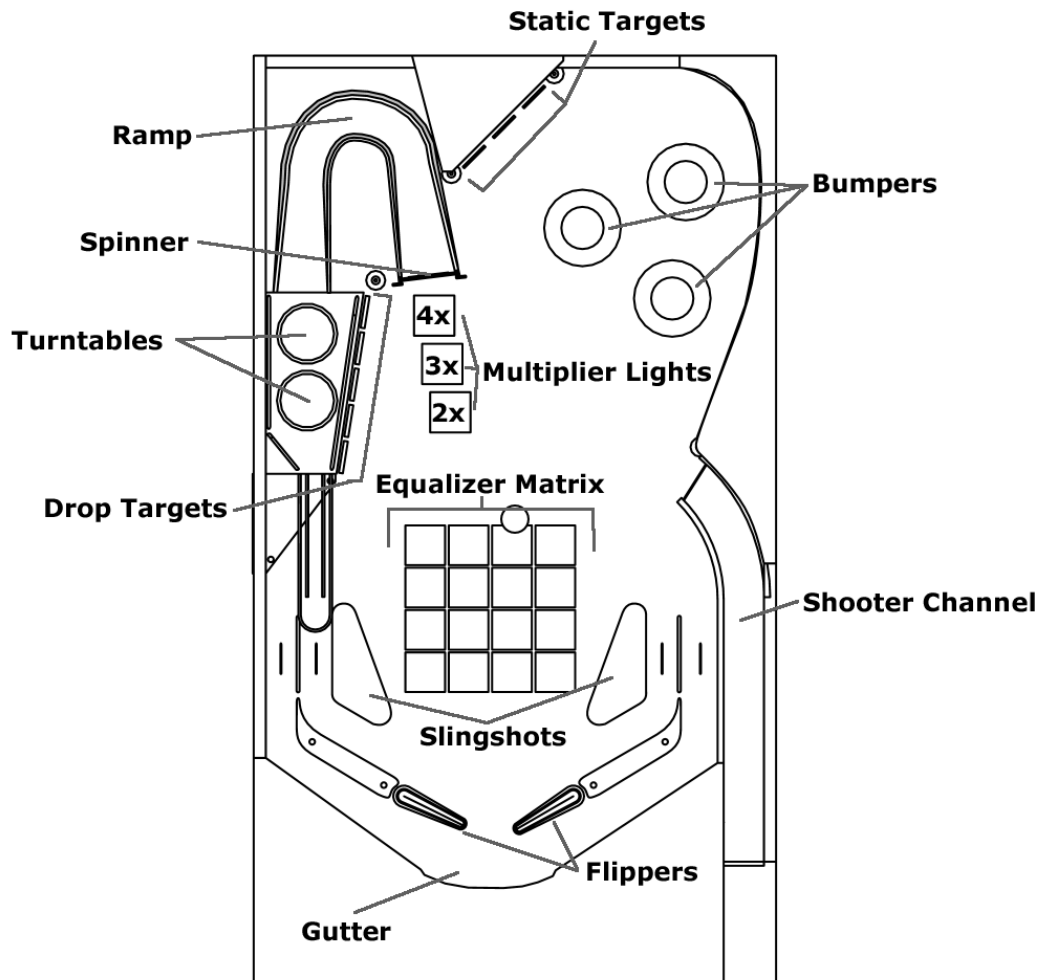


Figure 6: Playfield Elements

1. **Flippers** the flippers are used to hit the ball up the playfield. The flippers are controlled by the user using the flipper buttons.
2. **Slingshots** the slingshots activate when a ball makes contact with the flat inner surface and kick the ball away. The slingshots are worth 250 points.
3. **Shooter Channel** the ball is initially placed in the shooter channel. When the ball shooter is pulled and released, the ball is shot up the channel and into the playfield.

4. **Gutter** the gutter is the area at the bottom of the playfield. If the ball enters this area, it is lost and 1 ball is subtracted from the current ball count. If all balls are lost, the current game ends.
5. **Bumpers** the bumpers will bump the ball away when the ball makes contact with them. The bumpers will also light up on contact. The bumpers are worth 500 points.
6. **Static Targets** the static targets do not move but light up and provide 1000 points when hit.
7. **Drop Targets** the drop targets are targets that drop down into the playfield when hit with the ball. When all drop targets are down, a multiplier is earned and the drop targets reset. The targets will also reset when the current ball is lost. The drop targets are worth 500 points.
8. **Equalizer Matrix** the equalizer matrix is a set of surface lamps that change color based on the frequency response of the currently playing music track. These are purely a visual element.
9. **Multiplier Lights** the multiplier lights indicate the current multiplier level. The default level is 1x. When all the drop targets have been hit with the current ball, the multiplier level will increase by 1 up to 4x. While a multiplier is activated, all scores earned will be worth the standard score value times the multiplier.
10. **Ramp** the ramp provides access to the turntable area.
11. **Spinner** the spinner is an obstacle in front of the ramp that spins when the ball makes contact.
12. **Turntables** The turntables are obstacles that spin counter-clockwise when the ball makes contact. The turntables are worth 5,000 points.

4.2 Backboard Display Screens

The backboard display is used to let the player know important information about the game. When the game first starts up an animation advertising the game will start. This animation loops through in the goal to attract the attention of future players. This screen will remain displayed unless the player enters a coin or the player presses both flipper button at the same time. If both flipper buttons are pressed at the same time a high score list will scroll up the backboard scene. The high scores screen will disappear if the user releases a flipper button or the scores have finished scrolling. If a coin is inserted the backboard will move to the waiting to start screen that simply lets the user know how many credits are left and that the player should press start. When the player presses start the game will begin and the backboard moves to a purely information screen, showing credits, ball, and score information. When the game has completed the display will move on to a game over screen or a high screen depending on whether the score was high enough to make the list of top ten. See Figure 7 for a visual representation of this process.

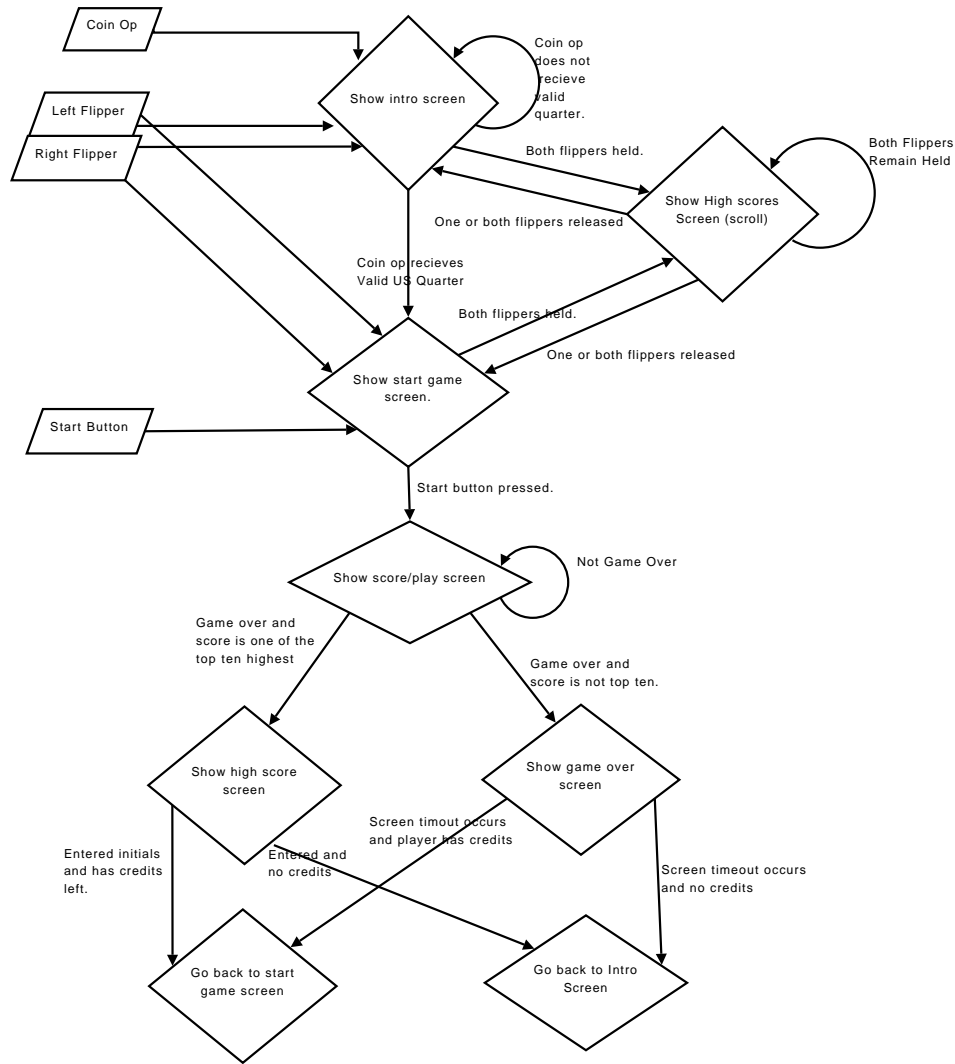


Figure 7: Backboard screen flowchart, showing the transitions of the screens through the game states.

5 Music System

5.1 Overview

DSP has an advanced music system that plays pinball sound effects as well as customized music. The music system keeps track of all sounds loaded, and handles playing them when the ball hits different obstacles that require a sound effect.

5.2 Custom Music Playlist

The music system allows the owner/operator of the machine to choose custom music that will play during the game. Music can be played from a CD or files located on the computer. The music

selection can be customized by modifying the playlist file that exists in Media/music/playlist.txt. The file has a simple format: each line represents one track in the playlist. The start of each line begins with either "cd" or "file" followed by a space character. For a cd track, it is followed by a number for the track on the cd to play. For a file track, it is followed by a relative or absolute path to a song file to play (path cannot include spaces). See figure 8 for an example playlist.

```
cd 4
cd 2
cd 3
file Media/mysong.mp3
cd 6
cd 7
file /home/mysong2.ogg
```

Figure 8: Sample Playlist File

6 Analytical

6.1 Power Calculations

$$EstimatedMaxUtilization = 80\% \tag{5}$$

$$EstimatedAverageUsage = 40\% \tag{6}$$

$$EstimatedAverageConsumption = 214W * 0.8 * 0.4 = 68.5W \tag{7}$$

6.2 Physics System (Newton) Input Parameters

The Newton physics system uses the laws and equations of physics to dynamically calculate all velocities, forces, and collisions of objects in the 3d scene. The use of this physics library eliminates the need to deal with these low-level physics equations, as they are computed behind the scenes by the physics engine. The source code of the library is not available, so it is not possible to determine the exact equations in use by the library.

The Newton physics engine uses the 3d shapes of the objects in the scene to calculate the physics, along with a mass assigned to each object. Figure 9 shows the parameters used for the ball and flippers as well as the gravity chosen in the pinball "world".

Parameter	Value
Gravity	-9.8m/s
Ball Weight	80 grams
Ball Diameter	1 1/16"
Flipper Weight	100 grams

Figure 9: Lifecycle and replacement cost for components used in the pinball machine.

6.3 Flipper Physics

One of the most difficult parts of the project was modeling the behavior of the flippers. The flipper physics is calculated by Newton, however the flippers are not free-moving bodies like the balls are. The flippers require extra physics constraints and behaviors in order to operate like real flippers.

To achieve the movement, a pin was added along the Y (vertical) axis at the pivot point to prevent the flippers from rotating in any other direction. The flippers are also constrained to the playfield floor so that they can't move in any direction. Constraints were added for the rest angle and max angle to prevent the flipper from rotating beyond these positions.

The spring return torque is applied to the flipper when the flipper button is not being held down. This simulated the force of a return spring on an actual flipper unit. When the flipper button is pressed, an activation torque is applied causing the flipper to rotate to the up position. This simulates the force of the coil on an actual flipper unit.

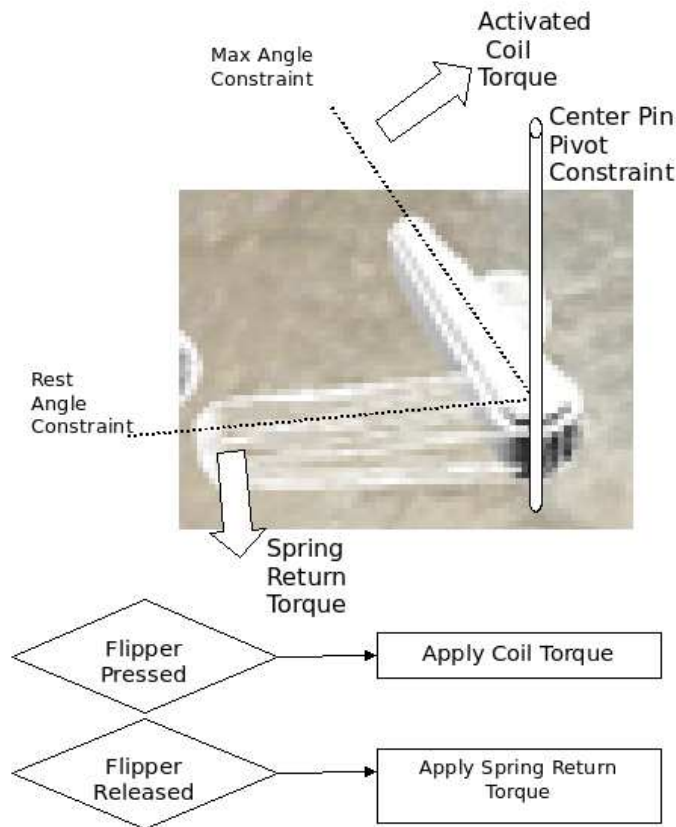


Figure 10: Flipper Physics

Figure 11: Flipper physics diagram, showing the joint and forces.

7 Software

7.1 Operating System

The general purpose computer that runs the simulation will use Linux since it has many open source solutions available and is much more embeddable and customizable than other alternatives.

- Modified version of Ubuntu Linux distribution Live CD
 1. Must fit on cd-rom (700mb)
 2. Must support graphics and program libraries (Ogre, Newton...et cetera)
 3. Must support hardware interface (ball shooter, coin op and input buttons).
 4. Support two monitors (one for score and one for main application)
- Boots directly into application
- Doesn't require a hard drive (lower cost, lower noise, and lower failure)
- May also be placed on a thumb-drive (if computer supports booting from USB)
- Allows software to be easily updated and also prevents corruption (read-only file system)

Ubuntu was chosen because of its excellent hardware support and its multitude of documentation available on the web. The project was designed for the Live CD option because it allows the operating system and pre-configured applications to boot on any machine without any installation of configuration of drivers. This provides fully plug-and-play compatible operation and allows hardware to easily be swapped without reinstallation of operating system and pinball software.

7.2 Software Overview

The software used in the project is based on a variety of free or open-source projects. Descriptions some of the software shown in Figure 12 are described below.

OGRE 3d Engine - <http://www.ogre3d.org/>

- Provides the 3d graphic rendering of the pinball playfield and backboard graphics.
- Integrates with Newton physics SDK to easily setup realistic 3d physics.
- Multi-platform (Windows, Linux, Mac)
- Widely used, supported, and documented

Newton Game Dynamics - <http://www.newtondynamics.com/>

- Real-time physics simulation (collision detection, dynamics)
- Will provide realistic behavior for ball, obstacles, flippers, ramps, etc.

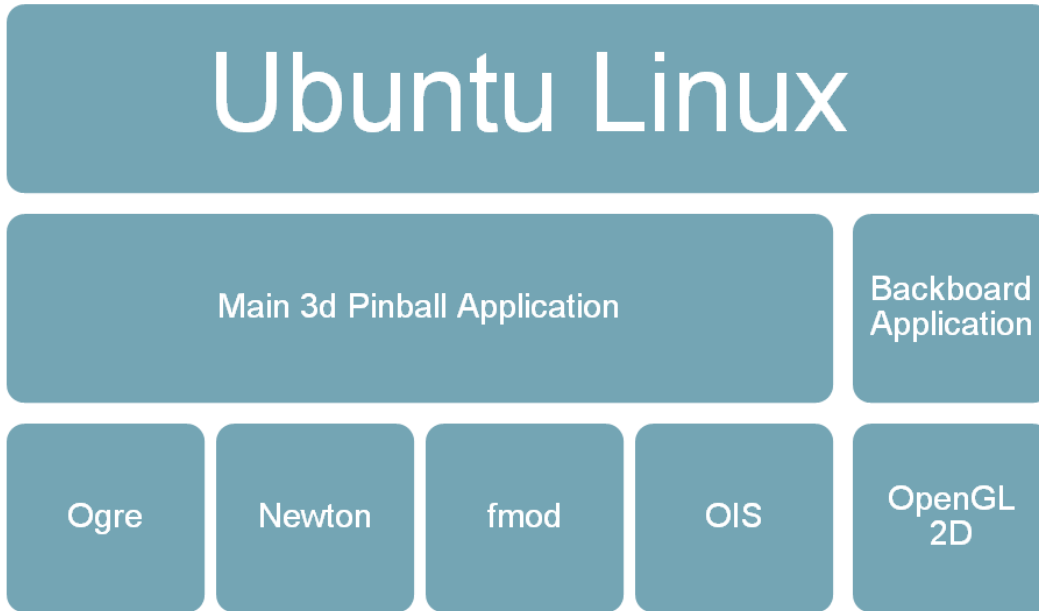


Figure 12: High level software overview.

- Multi-platform (Windows, Linux, Mac)

fmod - <http://www.fmod.org/>

- Music and sound effects system.
- Used in many commercial games.
- Multi-platform (Windows, Linux, Mac)

OIS - <http://sourceforge.net/projects/wgois>

- Object-Oriented Input System
- Multiplatform abstraction of input devices
- Handles input from keyboard, mouse, and joystick devices

7.3 Interprocess Communication

The backboard display application must run on a separate window manager from the main pinball application so that it can utilize the secondary display. This will require a method of inter-process communication(IPC) in order for the primary application to send information (graphics raw pixel data) to the backboard. Shared memory was chosen for IPC because it is included with the kernel, fast, low overhead, stores raw data and can be used as a simple pointer.

Name	Size	Description
control	1	0=Hold,1=Read
size	1	Stores the number of bytes stored in the data segment
data	20000	Holds the raw pixel data. When written to the size is stored first. Hold will be set to zero before writing and set to one on successful write.

Figure 13: Shared memory table.

Three segments of shared memory are needed for this project: control, size, and data. The first segment will be used as a simple control byte. If the value is zero the secondary application will hold (under no circumstances read or write into shared memory). If the control value is a one the secondary application will read the raw pixel data from shared memory and display it on the screen. In order for the secondary application to read the pixel data it first reads from the size segment in shared memory, which holds the amount of bytes stored in the data portion of shared memory. The backboard application will then read the proper number of bytes from the data segment and display it on the screen.

8 Performance Measurements and Testing Strategies

8.1 Performance

The project is used solely for entertainment value, but this does not mean that performance issues are by any means acceptable. Any loss of performance both detracts from the game as well as angers the user and the owner of the machine. For this reason it was important that the main pinball application stay above 30 frames per second at all times during play. To achieve this the music, sound effects and physics had to be optimized so that no action exists that causes the frame rate to drop noticeable lag in game).

Initial testing with the backboard application indicated that it required significant CPU usage to draw the simulated dot-matrix dots across the display. Since the backboard is meant to be secondary to the main pinball game, this was unacceptable since it could slow down the main application and interfere with gameplay. The first revision of this application used the SDL libraries to blit (copy the pixels) of the dot image to the screen. This would have to be performed once for each dot on the screen ($128 \times 96 = 12288$). Since this method used software buffers and the main CPU to do the drawing, it was very resource intensive. Running this application consumed approximately 40% of the CPU.

A new method for drawing the dot-matrix screen in the backboard application was developed that utilizes the hardware video accelerations to offload the drawing operations from the processor. This new method uses the OpenGL library to draw the dot matrix screen in 3d, but using an orthographic projection so that it appears to be 2d. Using the 3d hardware acceleration allows the dot image to be loaded into the video card memory as a texture only once, and the video card handles drawing the texture to the screen. Even the large name of dots (12288) is no problem for the hardware, compared to the millions of triangles/second of which it is capable. This new application consumes almost 0% CPU since all the work has been offloaded to the video card, a very significant improvement over the old application.

8.2 Testing

8.2.1 Controls/Inputs

A test application was created early in the project that tests the input buttons and the coin operator. The application consists of a GUI showing the buttons to test, a dollar amount and a meter for the ball puller. When a button is pressed the corresponding display element, a picture of a button with the label, lights up. When an official US quarter is dropped into the coin operator, the dollar amount shown on the test application should increase by exactly .25. When the ball puller gets pulled back the meter and percentage should increase. This application is very useful in determining that all inputs are working and that the infrared sensors on the ball puller are responding correctly. If any of the sensors were reading a faulty value, this application would be used to quickly locate the problem.

The input software used in the test application (OIS) is used in the pinball application as well. If for some reason it is this library causing the problem, it would appear to be a hardware issue.

To prevent wasted time, before any hardware diagnostics is performed, the operating systems built in control test will be used to verify that the input is not correctly interfaced with the computer.

8.2.2 Shared Memory

Originally the shared memory was tested by using the same test application designed for the controls. This shows that two applications can communicate. To test shared memory further test images were built using a graphics editor and that was sent to the backboard application to be displayed. This tests that the pixel transfer (data shared memory), size and control bytes all work correctly. If the backboard displays the test images sent over then the shared memory is being set up, written to and read correctly. If anything fails in the memory the shared memory wrapper will send an error message to the standard error stream that indicates where in the wrapper the error occurred and the error message returned from the kernel call.

8.2.3 Pinball Application

In order to fully test the playfield, keyboard keys were mapped to the hardware inputs of the pinball machine. This ensured that the keyboard matched exactly the internal process that the controls would. This enabled for testing to be done on the teams personal PC's. A special key was added that would insert a ball into the launcher without affecting any of the game variables. This made it easier to test some portions of the playing field as well as the backboard display scenes. There were a large amount of students willing to play the game, which resulted in some free testing and better coverage of different players techniques.

The GDB debugging tool was an important aspect in the testing process since it was used to debug parts of the application. The debugging tool was used to trace certain problems with the application and locate the area in code where the problem was occurring.

Memory management was an important aspect of the application since there are many resources being created and destroyed throughout the game. Testing was performed to ensure there were no memory leaks in the application which could cause it to crash after extended periods of operation. A suite of debugging and profiling tools for linux, called Valgrind, was used to detect possible memory leaks in the application. The Valgrind profiling tool monitors the application as it runs and reports any potential memory problems such as lost and unfreed memory. Using this technique, several problems were discovered and fixed that could have caused problems in the game.

9 Project Expenses

9.1 Manufacturability

The table showing the cost of each item can be seen in Figure 14. At first glance the cost of the project might seem quite large. It certainly is large in terms of a student project, but this was overcome by the donation of the plasma TV and the readily available computer and monitor. In terms of total cost, even with the plasma TV, it is less cost than that of a real pinball machine (\$5,000 USD). Authentic pinball machines must have each playfield manufactured and all the parts assembled. The digital pinball machine is not held back by these constraints, it is a one time

development expense and the hardware can be chosen, updated or downgraded at will. The game could easily be ported to PCs as well. As an added perk the digital pinball does not have as much mechanical parts, which are prone to fail at some point.

9.2 Item Costs

Item	Retail	Actual	Comment
Computer	\$500.00	\$0.00	Already available.
7600 GT Video Card	\$100.00	\$100.00	
42" Plasma	\$1,200.00	\$0.00	Already available.
15" LCD	\$180.00	\$0.00	Already available.
Coin Acceptor	\$25.00	\$25.00	
Flipper Buttons (2)	\$5 (\$2.50 each)	\$5.00	
Ball Shooter	\$15.00	\$15.00	
IR (tx/rx) (4)	\$14 (\$3.50 each)	\$14.00	
USB controller	\$15.00	\$15.00	
Wood	\$40.00	\$40.00	
Legs	\$75.00	\$75.00	
Leg Brackets and Bolts	\$40.00	\$40.00	
Plexi Glass	\$30.00	\$40.00	
Misc. Hardware and Paint	\$40.00	\$40.00	
Total	\$2,260.00	\$409.00	

Figure 14: Cost of items, both retail and to the project.

10 External Concerns

10.1 Safety Considerations

10.1.1 Epilepsy

”A very small portion of the population have a condition which may cause them to experience epileptic seizures or have momentary loss of consciousness when viewing certain kinds of flashing lights or patterns that are commonly present in our daily environment. These persons may experience seizures while watching some kinds of television pictures or playing certain video games. Players who have not had any previous seizures may nonetheless have an undetected epileptic condition.” [1]

10.1.2 Motion Sickness and Headaches

The pinball application uses a 3d simulation. The movement of the ball in the simulated 3d environment along with looking back and forth at the main application and backboard display could cause motion sickness or headache. If during play the user begins to feel ill or discomfort, the user should stop playing immediately.

10.2 Environmental Considerations

The pinball machine uses up a fair amount of power. To save on power the operator should make sure that it is unplugged whenever it is not in use. The TV and Monitor both have power saving capabilities. The power saving capabilities are left on and if possible the operator should leave them on.

11 Summary

The project was successful in creating a digital pinball machine. The feel of the game matches what one would expect from a pinball game, but was done in a new way. There were a few difficulties encountered with the project. The first was with the old ball puller design that used a linear potentiometer. In theory the linear potentiometer seemed like a good choice and initial trials looked promising. The problem arose with the final placement and connection with the ball puller. Every thing that was tried, although successfully transferring the pullers motion to the computer, looked risky. The design wanted to reduce the mechanical parts, not increase them. The new system with infrared emitters and detectors works much more comfortably. With the new system there is nothing physically attached, but the motion can still be transferred to the PC. Ensuring no contact with the ball puller means that replacement would only be necessary if the ball puller itself failed.

Another difficulty encountered was building the enclosure. Simpler options detracted from the appearance of the system and gave it a undeservedly cheap quality. The solution to this was relatively expensive, but resulted in a much better looking product. The construction took much longer than the original timeline estimated. Luckily the software progress made initially was enough to make up for the extra time taken to construct the enclosure.

12 Source Code

The source code for this project is too large to be reasonably printed out in this report. The full source code can be viewed on the project cd. The cd also contains additional documentation created with doxygen, a tool that formats the source comments in a easily readable form.

References

- [1] Nintendo - Customer Service Health and Safety Precautions.
(http://www.nintendo.com/consumer/manuals/precautions_console_pak_english.jsp).