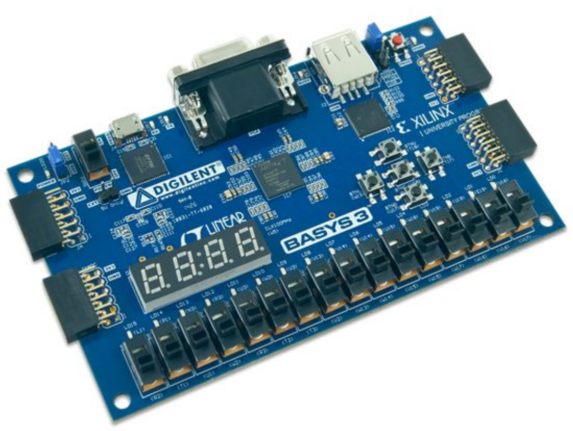
School of
Engineering &
Computing Sciences



Speed X3D on FPGA

Group 5:

Team Member	*Course*
Rajan Khullar	CSCI 660
Jeffrey Silva	EENG 483
Matthew Warshaw	EENG 483

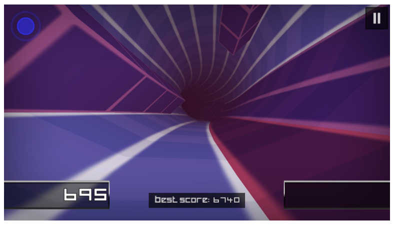
	Table of Contents			
1.0	*Introduction*			**2 - 3**
1.A		*SpeedX 3D*		**2**
1.B		*FPGA*		**2**
1.C		*Programs*		**3**
2.0	*Planning*			**3**
2.A		*Layout 1*		**3**
2.B		*Layout 2*		**3**
3.0	*Implementation*			**4 - 6**
3.A		*Pulse Divider*		**5**
3.B		*Clock Box*		**5**
3.C		*Vga Sync*		**5**
3.D		*Frame Box*		**5**
3.E		*Segment Display*		**5**
3.F		*Core*		**5**
3.G		*Canvas*		**6**
4.0	*Appendix*			**7 - 9**
4.A		*Circle*		**7**
4.B		*Code Snippet*		**8**
4.C		*Controls*		**8**
4.D		*Contributions*		**9**

1. Introduction

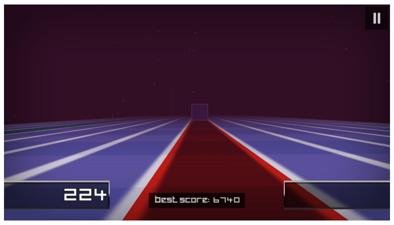
A. SpeedX 3D

SpeedX 3D is an app available on both Android and Apple devices. The game has an arcade style look and has several features and levels. For normal gameplay there is only one control mechanism for the user. The player holds their device so it is oriented landscape and then uses it like a steering wheel. The gyroscope of the device detects the orientation. The app reads this data in order for the user to move left or right.

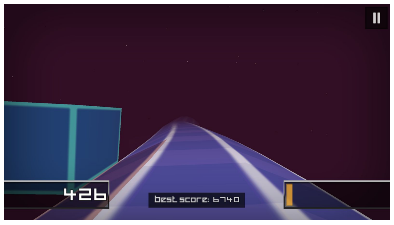
The layout of the game is complex. The player is actually not shown on the screen. Instead the player is a camera point of view. More along the lines of a FPS game. Throughout gameplay the base underneath the player changes curvature. The base progresses between three modes. The first mode being the inside of a tube; second is flat, and on the third day the player lies on the outside of the tube. The modes cycle back and forth linearly, which always results in smooth transitions. As in the base cannot go from inner tube to outer tube without first becoming flat, and vice versa.



Inside Tube



Flat



Outside

B. FPGA

The Basys 3 is the Field Programmable Gate Array development board provided by NYIT. An FPGA allows digital logic designers to create and test designs without manufacturing a chip. The alternative is to design an Application Specific Integrated Circuit (ASIC) for each iteration of a design. Since ASICs are very expensive to manufacture, cleary FPGAs have a tremendous advantage. The benefits of using an ASIC over an FPGA, such as improved power efficiency, don't apply to this project since we can always rely on laptop power. FPGAs primarily consist of logic units, channels, multiplexers, and registers. To program an FPGA the user sends the device a bitstream which sets the values of all the registers. This in turn controls the multiplexers, which program the circuit. The circuit does not change shape or form, but the behavior changes since the multiplexers select which lines can pass a value to a channel.

Another advantage is scalability. Since the primary components are an array they can placed very efficiently just like Random Access Memory (RAM) to optimize space on the silicon. The result is having a circuit as small as possible with a number of components. The physically smaller circuit means that values can propagate really fast as opposed to a larger circuit. The less delay also means that the speed limit for the system clock can be increased, which allows for even faster processing speeds.

C. Programs

The edge between Hardware and Software becomes blurred with FPGAs. Hardware refers to the physical circuits inside a device. This includes components such as silicon, transistors, et cetera, all the way up to digital gates. Software usually refers to programs that are executed by a Central Processing Unit. With an FPGA, the program in question is not executed by a CPU, but still alters the behavior of the device.

2. Planning

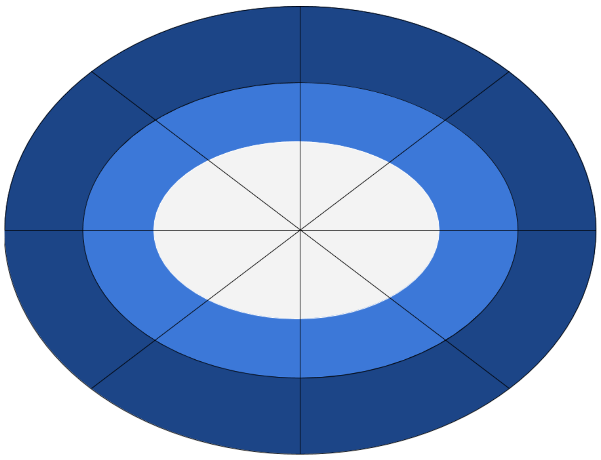
While planning how to implement SpeedX 3D we had two major ideas for the basic layout of the game. The first was to use the mode of the game where the player is inside of the the tube. The second was for the mode when the player is on top of flat ground. We decided to use the second layout since we not able to get far with planning the first one.

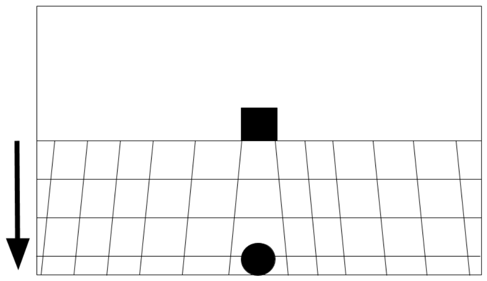
A. Layout 1

For the first layout, the tube would be drawn using concentric circles. Diagonal lines would be drawn through the center of the tube. In order to simulate movement through the tube the radius of each circle would modularly increase. In order to give the player feedback for moving left or right each the direction of diagonal line would be offset by some modular degree. This design is problematic since the lines would need to use polar coordinates and drawing those involves complex math. We would have needed to use floating point numbers as well as sine and cosine.

B. Layout 2

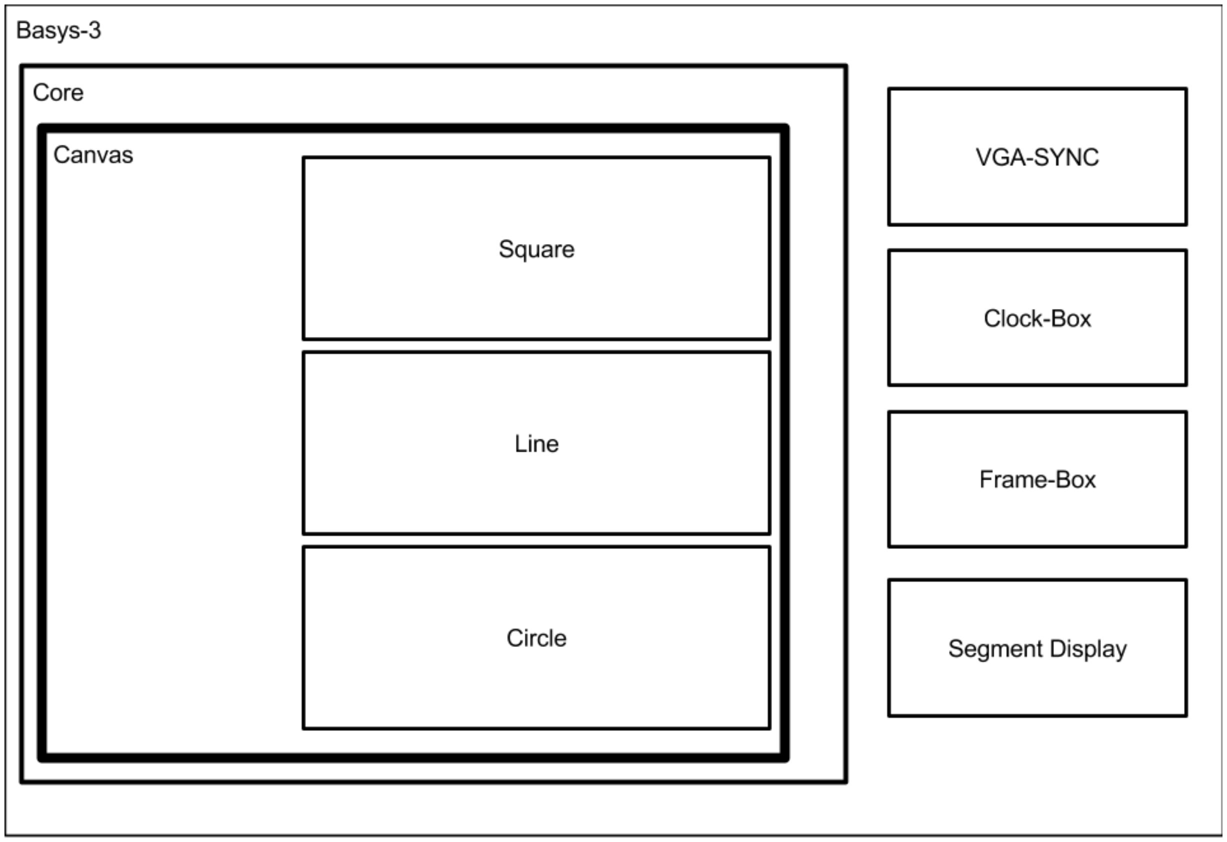
In the second layout, the screen is split into a top and bottom half. The bottom half contains the majority of the game mechanics. Whereas the top half helps in providing a 3D feel. The player is drawn using some object such as a dot or a square. To simulate movement horizontal lines would move down towards the player. To add a 3D effect, vertical lines would also be drawn statically but they would be slightly slanted inwards (bottom to top). The monster or obstacle of the game could be drawn from a ROM image (sprite) or just a square that grows in size. The monster would move towards the player and once the player and monster touch each other the game is over.


Layout 1


Layout 2

3. Implementation

The system design for our project is very hierarchical. At the top level is the actual Basys3 board. In this module we describe all the input and output ports used in our *program*. As for the architecture inside of the top module, we opted to strictly use port maps and signal declarations. In other words the top module consists of several smaller modules connected together to some wires and ports. The smaller modules primarily include components such as pulse dividers, vga syncs, cores, etc.



System Architecture

A. Pulse Divider

The pulse divider component is responsible for dividing an input clock by some number to generate a new clock. In order to achieve this a count variable is used. On reset the count becomes zero and the output clock becomes low. Then on every rising edge of the input clock the count increases by one. When the count reaches a limit, the output clock signal is inverted. In order to be reusable this component must be generic.

B. Clock Box

The Basys3 FPGA standard clock speed is 100MHz if we wanted to change how an object behave in respect to time then we use the system clock and divide it. Frequency is f=1/T, or a cycle per second. We first find how many seconds is in a cycle and change our equation to the amount of seconds we want our object to behave in terms of cycles. The clock box simply wraps up all the frequencies into one generic package.

C. VGA-Sync

This component is responsible for tracing a standard monitor with 640 by 480 resolution at 25 MHz. The process is complicated since it involves values outside the target range. So it is abstracted to the rest of the program by outputting the current x and y. It also signals when the trace is on screen and when one frame has been completed.

D. Frame Box

The frame box is similar to the clock box in the sense that it wraps up some frequencies. The difference is that the input clock here is actually the end of frame signal from VGA-Sync. When doing updates on the screen it is critical to perform each update while the VGA is not drawing. If the update was performed while the drawing occurs then the image on the completed frame would be incorrect. The result is a blurry or glitchy game.

E. Segment Display

The seven segment display on the Basys3 can be used to keep track of numbers. The vhdl code for this is not very complicated. Nonetheless, we opted to abstract the circuit to the rest of the game. This component simply reads a 4 digit hex number and displays it on the board.

F. Core

This is the most important module for the project. The core contains variables such as sprite positions and sizes, as well as other components. There are several processes to control how the state of the game transitions from start to finish. First there is a timer that delays the movement of the monster and player for some time after reset. The purpose of this is to ensure the player can see the screen and is ready to play the game. Another process is responsible for updating the positions of the player and monster. Some factors involved include whether or not the player is hitting the monster, and

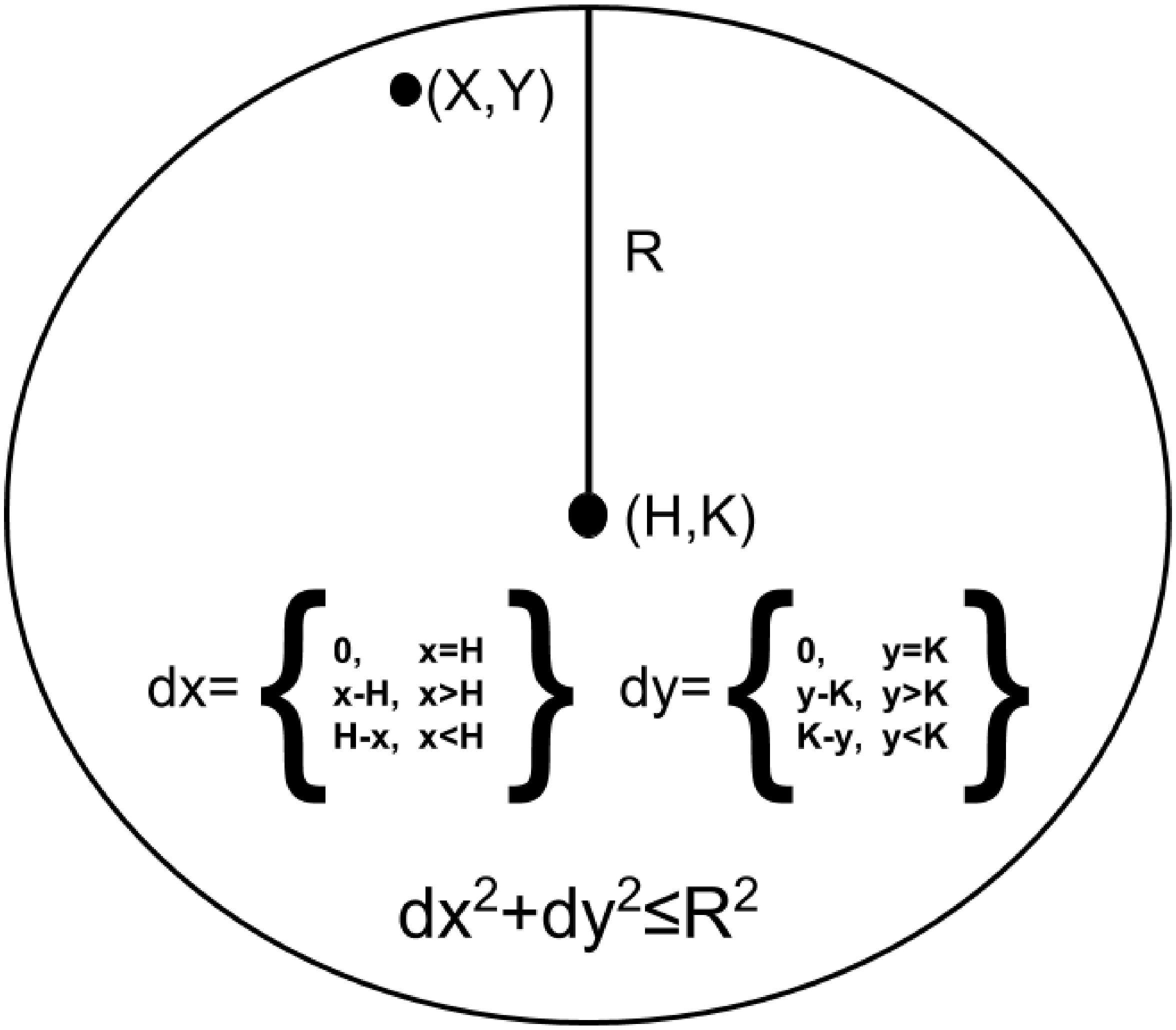
whether or not the player is touching the screen border. Other less critical data includes the color of the background.
All of these variable come into play in the module and that is why the core is most important.

G. Canvas

The canvas takes in the values from the core including current_x and current_y. The primary output of this module is which color to draw, if any. The canvas contains several other modules as well but all of them are related because they are geometrical objects. In our implementation we have layers on the screen including ground, circle, sky, player, monster, border, etc. Essentially the canvas forwards the coordinates to each geometric object and receives whether that object should be drawn. Then precedence is applied to layer the objects on top of each other. For example, if the player and a line both should appear at some coordinate, then the player should be drawn over the line since it has higher precedence.

4. Appendix

A. Circle



$$dx=\left\{\begin{array}{ll} 0, & x=H \\ x-H, & x>H \\ H-x, & x<H \end{array}\right\} \quad dy=\left\{\begin{array}{ll} 0, & y=K \\ y-K, & y>K \\ K-y, & y<K \end{array}\right\}$$

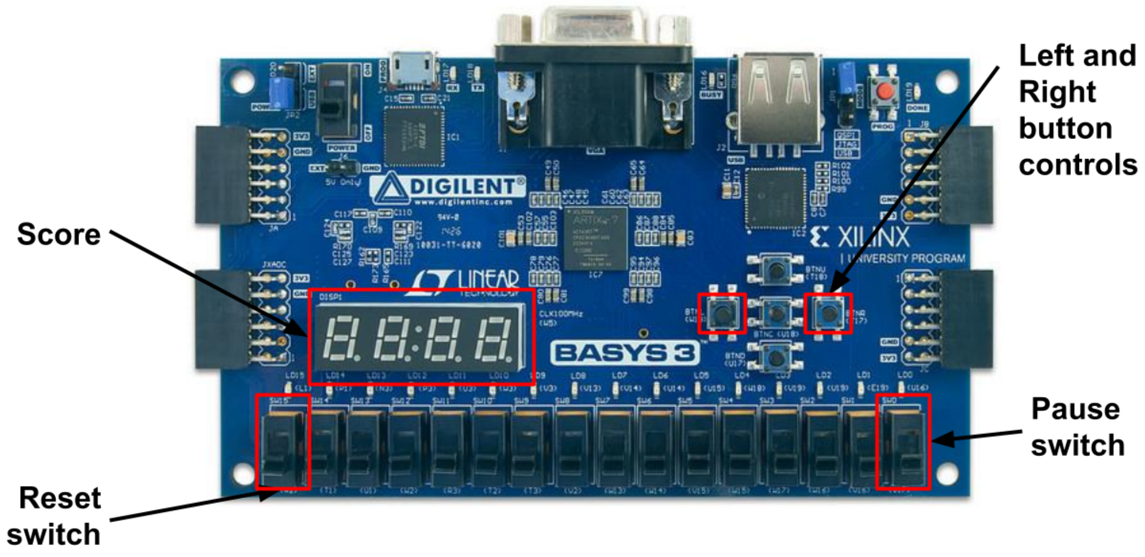
$$dx^2+dy^2\leq R^2$$

How to Draw a Filled Circle

B. Code Snippet

-- Update Sky
process(rst, clk_60f)
begin
  if rst = '0' then
        sky_addr <= (others => '0');
 	
  elsif rising_edge(clk_60f) then
        if sky_addr = x"3e" then
        sky_addr <= (others => '0');
        else
        sky_addr <= sky_addr + '1';
        end if;
  end if;
end process;
	This process from the core is used to update the background color of the sky every 60 frames. The sky address is incremented until the maximum address of the ROM is reached.

C. Controls



D. Contributions

	Rajan Khullar	*Jeffrey Silva*	*Matthew Warshaw*
Planning	✔	✔	✔
VHDL - Core	✔	✔	
VHDL - Canvas	✔	✔	
VHDL - Misc	✔		✔
Graphics		✔	✔
Report	✔	✔	✔
Slides		✔	✔