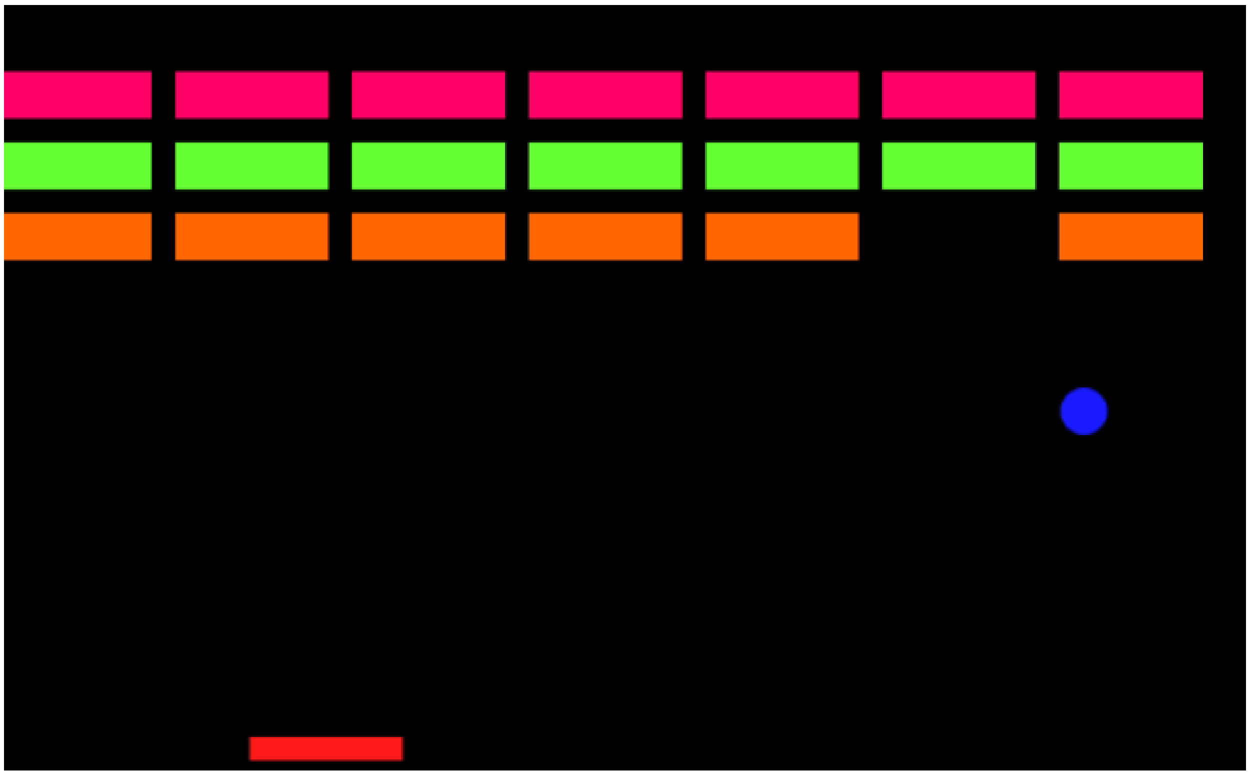

ENSC 452 Project Proposal

Brick Breaker

01.22.2018

Ruisi Wang (Sam) - 301154103

Jianglin Fu (Arlene) - 301256171

Simon Fraser University

Table of Contents

1.0 Introduction

The goal of this ENSC 452 course is to implement a program on ZedBoard with both Xilinx's system software and hardware. We decide our project to be a video game similar to brick breaker. In our game, user can move the bar at the bottom of the screen left and right in order to bounce the movable ball up and try to hit the bricks. Once the ball hit any brick, the brick will be downgraded and eventually wiped out. We will split our job into 6 milestones and using agile methodology to accept greater flexibility and to get higher efficiency. Each part will be tested by specific test plan to ensure the completeness.

2.0 Background

Breakout is a traditional arcade game developed and published by Atari, Inc. Brick Breaker is a clone of Breakout. The game consists of 50 bricks on the top of screen, a flying ball and a movable paddle on the bottom of screen. The paddle is controlled by user to move either left or right. Player needs to use this paddle to prevent the ball from falling out of screen. Three unbreakable walls on the sides of screen are used to deflect the ball. Collision between the ball and bricks can make brick downgraded or disappear. The speed of the ball gradually goes up with time. The level of bricks is also upgraded as time goes on, which means it requires more than one collision to smash the upgraded brick. The game will continuously proceed as long as the ball is not falling out of screen. During playing, user is able to pause the game and continuous afterwards.

This game connects with a standard keyboard to get user input and use VGA port to display the whole video screen. If we have enough time, we will play audio and make sound be synchronous with breaking a brick.

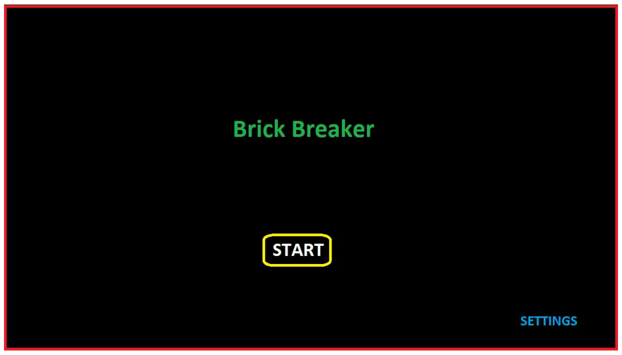

Figure 1: Example game start screenshot

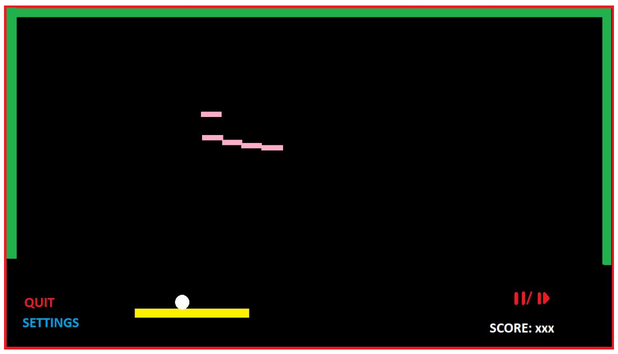

Figure 2: Example game play screenshot

3.0 System Overview

3.1 Use Case Diagram

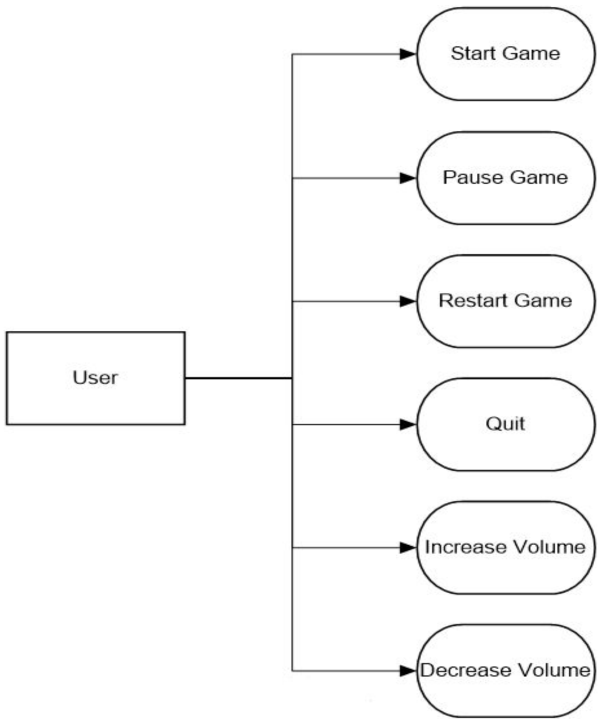

Figure 3: Use case diagram

3.2 System Block Diagram

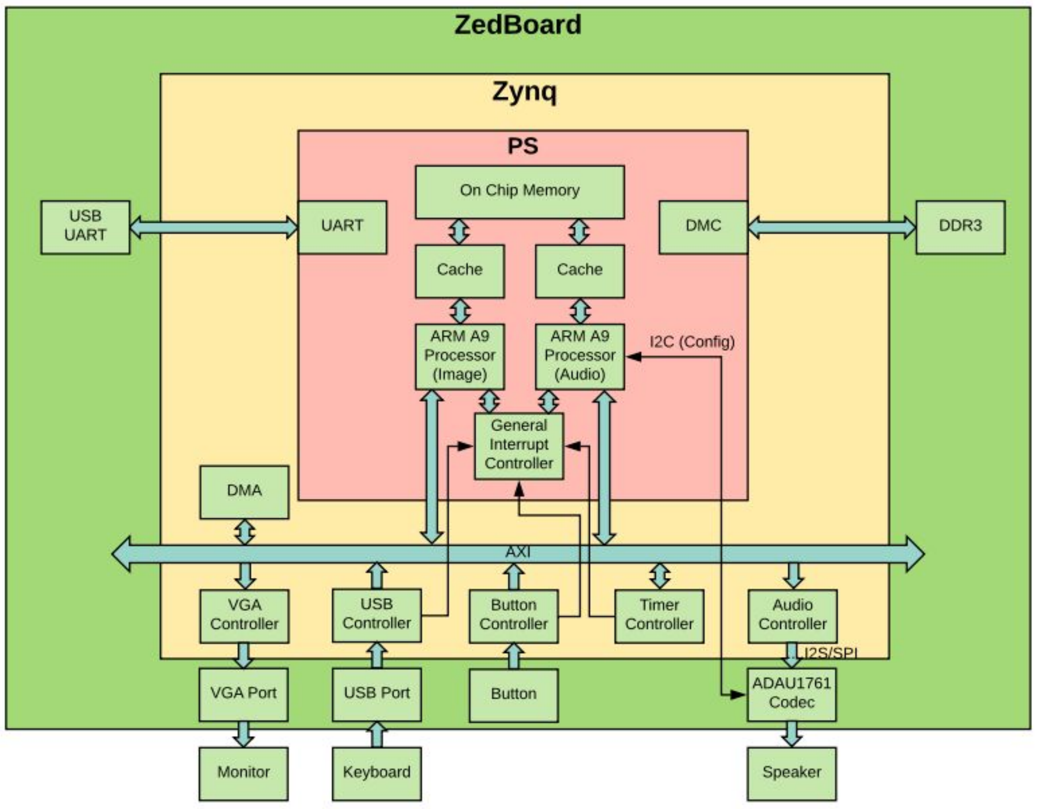

Figure 4: System block diagram

3.3 Algorithm Flowchart

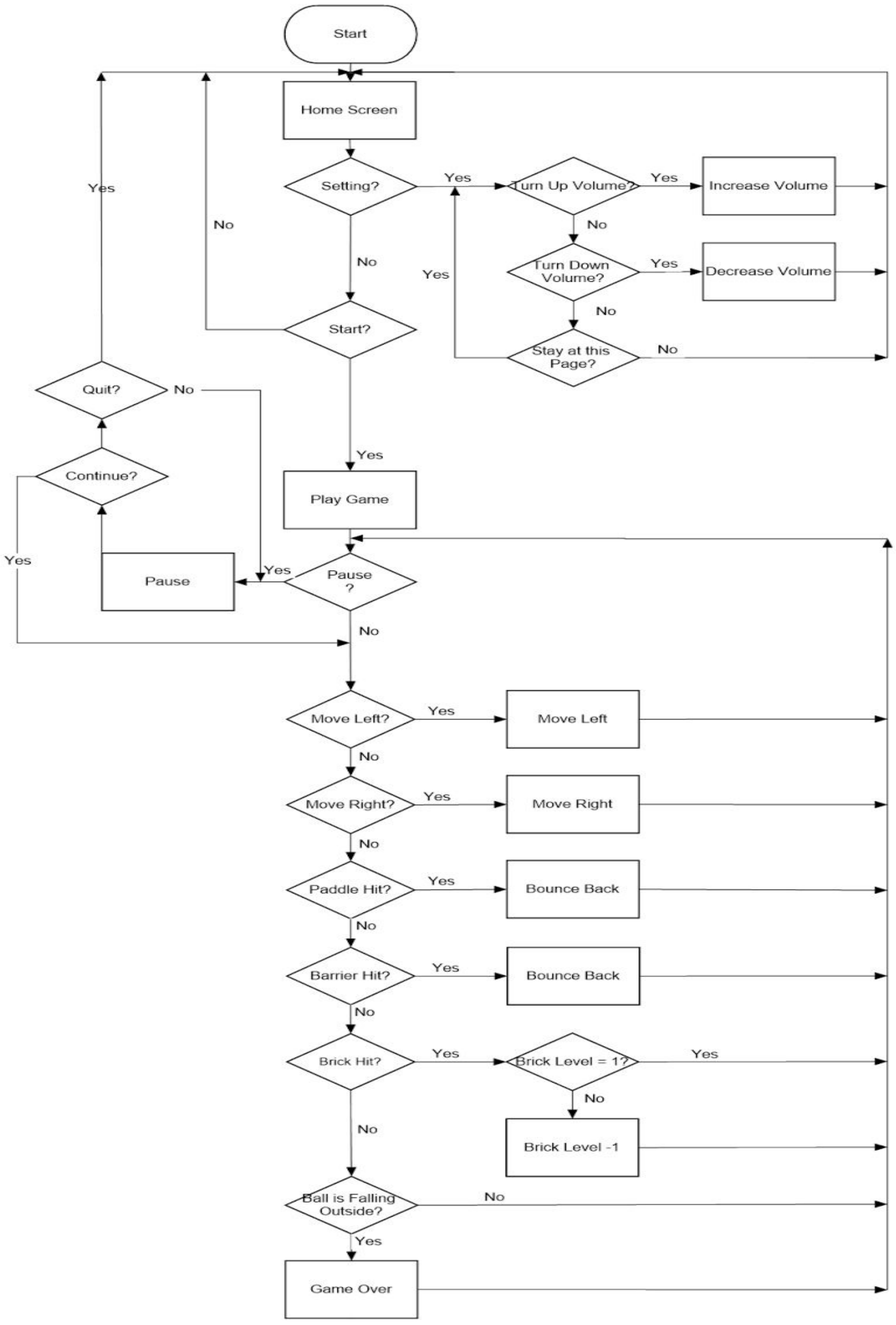

Figure 5: Algorithm flowchart

3.4 Design Methodology

We have divided the whole project into six milestones. We will design each individual parts and components which the game will need. By using this method, we will be able to break down the whole task to several different smaller tasks and each of us can work individually and putting our work together will be convenient as long as we agree on the connection agreement. Another advantage of dividing the task using bottom up design methodology is the ease of testing. Modular design allows us testing each small task's result and debugging each small task will be rational and well-founded.

Arlene:

Splitting the project into small pieces ensure us to finish work more efficiently and more motivated.

I decide to work on VGA first, since the ability to display image on monitor is the base of our game. Previously I had experience showing image on a monitor through VGA while running UART driver on PetaLinux. This time I will need to figure out the way implementing similar stuff on Zedboard. Once I am able to print pixels on the display, I will try to implement an interrupt from a push button. Defining deflected path of the ball, varying brick level and ball speed should be done accordingly.

The game logic definitely could be improved by adding more specific rules. Since the program is written modularized, it is easy for us to add more new features if we have extra time. For example, we can use mouse to control the whole game instead of keyboard or button. We can keep historic playing record, which allows to view the highest score, save the game and reload the whole game. Besides, if we complete all the targeted features, we can increase resolution of the game by using DMA to access the DDR3 through AXI bus.

Sam:

I will be responsible for the audio which will be stored in the memory and playing it to the speaker when needed; keyboard input which will be connected to the board by a USB controller, and synchronous both image and audio at the end which is the key to the whole project and the best way to increase user experience.

Using step by step planning, I will break down my tasks into smaller parts which will be each milestone, using bottom-up method, I will implement each task by research on the topic and learn from the tutorials giving. Each my milestone will be built based on the previous one, so being on time and stick to the plan will be the key to reach the final success of the project.

The game will have sound effect and user will control their brick by the left key and right key from the keyboard. Therefore, using my method described above, it will be the best way to achieve the goal.

3.5 Requirements

The following list is ordered from the highest priority to lowest.

1. The game should operate with no error
2. The game should be easy to get started
3. The game should have background sound playing during the whole game

4. The game should display all movements and pixel changes smoothly with no delay
5. User can control the paddle with keyboard
6. User can set the background volume from 0 to 10
7. User will be able to pause the game and resume the game afterwards
8. Background audio should be synchronous to the game, which means playing different sound towards corresponding smashed brick level
9. User can control the paddle with mouse

3.6 Constraints

The following list is ordered from the highest priority to lowest.

1. The game should have a high refresh speed to ensure the appearance of fluid motion
2. The game should be able to perform collision detection fast enough between the ball and walls, paddles or bricks before the ball moving to next location

3.7 Preferences

The following list is ordered from the highest priority to lowest.

1. We choose our game as portrait mode instead of landscape mode
2. In order to improve the game appearance, we define three unbreakable walls around the brick stack. User can clearly see the ball has been deflected when reaches the screen edge
3. The reflecting angle of the ball only depends on incident angle; speed and acceleration will not affect reflection path
4. The highest score will not be available on our initial game version

3.8 Learning needs responsibility

Arlene:

The first thing for me to do is doing some research about using VGA controller to connect the display and both programming logic and on-chip memory. Since the VGA connector on the Zedboard is connected to the PL part of the Zynq, I need to design or find VHDL code for a VGA interface and a matching driver. Previously I was thinking use a VGA DAC circuit to deal with stuff like hsync, vsync and RGB. However, there is no DAC circuit on Zedboard, I need to search for another way to make this work, for example, add a DAC interface with PMOD.

When display image on the monitor through VGA controller, I need to be familiar with how DDR3 works and how to increase the efficiency of displaying the whole game on the monitor. From the lecture, I think I can investigate the principle and usage of double buffer. This may help me find out the way to improve performance and save more power.

It is also very important for me to fully understand how to deal with user interrupt. It is not quite clear for me how to receive interrupt from hardware and handle it in software. In other words, after figuring out the way to handle interrupt, I have to understand how to let programming logic reacts to the interrupt accordingly as well.

In terms of memory access, the other thing to search for is trying to understand the way to access PL memory from PS on Zynq/Zedboard. Besides, our project also need to access data from DDR3 of PS side from PL part.

Sam:

Completing my milestone in this project will be challenge.

I have to learn audio input as bitstream which I have no experience. I have to create another processor just for the audio output and future logic. another thing I have to learn about is using usb port to implement keyboard as a input hardware in our game which I have never done before, the mapping and driver part of the game could be quite different then the VGA.

In order to learn all this, I have to research on the audio example and data sheet for the Zedboard and create corresponding gates and port in the hardware designing part of the project, connect all the audio constrain will the system bus, and getting correct input from the "Setting" of the game.

By using and learning both needs above, I will be able to send user input to the game which will change the position of the brick and inrich the game to have audio response.

4.0 Milestones and Testing Plan

After evaluating and discussing each team member's strength and available resource, we have come up with our each milestone which plan with reasonable time limits and each member's ability as showing below. All work and milestones will be done each week as individual, but all tasks will be combined as a team and any debugging or testing will be teamwork and helping each other is also our responsibility.

As shown below, a brief overview on the milestones of the project for each member.

4.1 Milestone 1

Arlene

Goal	Display image through VGA
Implementation	VGA controller should read from memory, then use one processor to modify the display
Purpose	This step will allow us to display the game on screen in the future
Demonstration	3 side unbreakable walls, the ball and the paddle will be displayed on screen, currently all the images are static
Test Plan	Manually see declared images on the screen through VGA

Sam

Goal	Able to play sound
Implementation	read data from the memory and play it to the audio jack
Purpose	This step is necessary to begin the sound implementation logic to the game
Demonstration	a sound will be play to the speaker
Test Plan	Manually test sound to the speaker

4.2 Milestone 2

Arlene

Goal	Make BTNL/BTNR to control the paddle and ensure it is not cross the boundary of screen
Implementation	Connect the AXI GPIO and button on the board. Every button pressed should send an interrupt, and the screen should immediately update the pixel change of the paddle.
Purpose	To upgrade the screen from static to dynamic. The paddle must be able to move since this is one part of the game.
Demonstration	In this stage, the paddle is able to move and controlled by BTNL/BTNR. The paddle should start moving from the bottom center. The speed is constant and paddle only can move horizontally. BTNR represents moving right, BTNL represents moving left. When paddle reaches the end of two sides of wall, it cannot go forward.
Test Plan	Press BTNR in order to move paddle to right, and then press BTNL to make the paddle to left. Next, keep press BTNR until the paddle reaches the screen edge to ensure the bar is not going outside of screen.

Sam

Goal	Being able to set volume up & down from the screen through VGA and set the volume limits
Implementation	Getting logic signal from the screen and apply it to the audio jack and changing the value of the waveform as the sound played, this will require access of the memory and pass volume data from the screen to the audio jack.
Purpose	Being able to change the volume so that the user can choose and have more control of the game
Demonstration	A sound will be play to the speak and volume will be change from 0 to 100 randomly to see if logic are correct
Test Plan	Manually test 3 different sound format at different times, test the speed that between the time when the user change the volume and the time the volume actually change at the speaker

4.3 Milestone 3

Arlene

Goal	Make the ball have ability to move anywhere inside of the boundary, following specific path.
Implementation	The initial position of the ball is on the middle part of paddle. In this case, I assume every side of screen is non-breakable boundary. I need to define some straight line function (with different angles) and enable the ball moving along each function. I also need to set that, the ball starts to move after detecting the movement of the paddle.

Purpose	The moving ball is the key of this game. This step is the prerequisite for us to set the moving direction of the ball.
Demonstration	In this stage, the ball starts from the middle of paddle, it should move upwards along function y=kx+b. Every corresponding pixel should change simultaneously.
Test Plan	Writing code to ensure the ball can move straight up automatically

Sam

Goal	Make the paddle be able to be controlled by keyboard and move only in the horizontal direction
Implementation	using hardware access in the hardware design of Vivado, implement the keyboard in Vivado by mapping <- and -> key into our design and access them in the logic
Purpose	By enable the keyboard, we will allow our game to have much more different feature and access in the future, and increase user experience of the game
Demonstration	A terminal will be open and each time a key is press on the keyboard will result in reaction on the terminal, and each reaction on the terminal will clearly shows it is the corresponding key that is being press
Test Plan	Manually test the keyboard will hands and test each key with different time that is pressing

4.4 Milestone 4

Arlene

Goal	The ball should be bounced back when hit the boundary or paddle. If the ball falls out of the screen, game over. (Detect collision)
Implementation	I need to keep track of the ball location using (x,y), generate a function to detect whether the ball and the paddle are in the same place or not. If the answer is yes, then the ball should be bounced back along a path, which is predefined. If the ball location is out of screen, then freeze the screen.
Purpose	This is the main action that can make the ball keep moving.
Demonstration	Once I control the paddle move to right (or left), the ball is considered to move (the ball has been triggered). During the movement, as long as the ball contact with walls or paddle, it should be bounced in the opposite direction.
Test Plan	Manually run the game and to see whether the ball goes straight through the barrier (wall and brick) or is bounced back. Also need to manually control the paddle do not catch the ball, to see whether the game freeze or not.

Sam

Goal	Brick disappears when hitted by a ball in the game play

Implementation	By access the C code in the implementation, I will implement the logic which the brick information will be discard when the ball's position hit the brick position in any direction, and when the buffer try to draw the next picture for the next cycle the hitted brick will not be in the list.
Purpose	This is the main game play logic for our game which is the only way for the player to win: by bitting every brick in the game field
Demonstration	By this point our game can be play for some time, so the tester will be playing the game in some way and hitting the brick will result destroying the brick and winning
Test Plan	An auto game will be played and each brick will be hit to see there is any issue.

4.5 Milestone 5

Arlene

Goal	Vary speed of the ball and brick level
Implementation	Increase the counter once the brick has been hit and record the last smashed brick's location. If the counter reaches 10, add a new upper level brick to last brick's location. If the counter reaches 30, increase the speed of the ball. (We need to reset the counter after performing the above action). Each brick should have an attribute/flag about its level. And each time the brick has been hit, decrease the level by 1.
Purpose	It is interesting to increase the gaming difficulty step by step.
Demonstration	All the initial bricks are in level 1. User control the paddle to move, after breaking ten brick, one level-2-brick will appear in the position of the 10th smashed brick. After 30 bricks has been hit, the ball speed will increase.
Test Plan	Write code to change the counter for smashed bricks directly to 9 and 29, respectively. Then start the game, to see new generated brick with different pattern and higher speed moving ball.

Sam

Goal	Pause game play, resuming game, quiting the game from any stage
Implementation	Using c code to stop game cycle in order to pause game, enable game cycle, giving reaction, and ending the game. Implementation is required, and the logic should be down in the software
Purpose	To let user have more control of the game
Demonstration	Game will be played and paused, resumed, and quit.
Test Plan	Manually test these functions and test them in each stage of the game.

4.6 Milestone 6

Arlene

Goal	Display score and enable home screen
Implementation	Store the number of smashed brick into memory, and display it on the screen. Also create the home screen which contains setting menu and play button.
Purpose	Deliver nice GUI.
Demonstration	The home menu should have setting button, which allow user to change audio volume. It also has play button to start the game. Once start the game, the game interface will show a real-time updated score information.
Test Plan	This can be tested by starting the game and it should have a main menu. After starting the game, every brick user crashed, the score will increase by 1.

Sam

Goal	Synchronous sound with ball
Implementation	Using mutex and lock to Synchronous the audio and the game, to have sound effect
Purpose	To enrich the gameplay experience
Demonstration	Game will be played at this stage and hitting the brick will now result in sound effect
Test Plan	Brick will be discard in different stage of the game to see if sound will always be play.

4.7 Final Demonstration

Goal	Integrate all the milestones to have a smooth and complete game
Implementation	The game will be implemented in code with zedboard, it will contain features specified in each milestone and will be a complete product
Demonstration	When user start the game, the home page will show up first. User is able to click start button to start the game, or go to setting page to adjust background volume. In the beginning of the game, there are 50 bricks in the middle-upper part of the display, three unbreakable side walls indicating the edge of game interface, a paddle in the bottom and a ball on that. During the game, player can use left/right arrow on the keyboard to control paddle's moving direction. The ball will keep reflecting as long as the ball encounter a barrier (brick, wall, paddle). If the paddle fails to catch the ball, the game is over. Otherwise, user can keep playing infinitely. Player is also able to pause the game at any time and resume it.
Test Plan	One will be playing the game and experence with the game as a user.

5.0 Design Concerns

Arlene:

The biggest concern is whether we can finish the whole project on time or not. We only get two months and there are so many things to learn, this could be a huge challenge.

Aside from the time factor, interrupt handling is also a big issue for us. Ideally, once user send an interrupt, I should be able to catch it and start changing the paddle location. In the meanwhile, I should using the updated paddle location to perform collision detection simultaneously. In other words, while dealing with interrupt, the ball are still moving and each brick has potential to change. Therefore the frame updating rate should be high enough to ensure the smooth of the game.

Sam:

There are three different parts of the concern case I will be mainly concerned about.

For the Audio, no doubt the Synchronous time constraint will be a big issue, how fast can the sound be play after it should be play will be the succeed / fail of our game. we will be testing and improving our game experience as good as possible, and adding more sound effect if there is time.

Another thing about Audio implementation is the multiple sound effect at the same time. if the the ball touches the brick as “ball sound” is playing at the same time, should we stop the sound and play it again, or should we play the second one as the first one is playing? the answer should be the second one, but doing this might cause problem that we do not know yet which is what we should look out for.

Last but not least, when we mapping the keyboard into our project, we should consider about the multiple input and the wrong input from the keyboard. our design choose will be ignored the wrong input and letting one input for each available cycle, and these corner case should be carefully tested.

Above design concerns should be test but not limited to these three concerns. more will be be add to this list as the project goes on, and each of them should be test in the test plan.