VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

DO NGHIEM AN

RAPID ROLLER: A GAME FOR WINDOWS PHONE

Telecommunications and Technology
2012

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
TELECOMMUNICATIONS AND TECHNOLOGY

ABSTRACT

Author	Do Nghiem An
Title	Rapid Roller: A game for Windows Phone
Year	2012
Language	English
Pages	61
Name of Supervisor	Ghodrat Moghadampour

In 2008, Microsoft stopped developing the "Windows Mobile" operating system and created a totally new mobile operating system "Windows Phone". Windows Phone supports multi-touch and multi-tasking which provides smooth user experiences. As a new operating system, Windows Phone has gained a lot attention from developers. Its applications have been growing rapidly. Game application is one of the most growing applications. As a result, game industry becomes more valuable and potential.

Game development is a process of combination of coding skills and designing skills. The most important part in game development is the game concept. It should be simple and addictive.

The goal of the project was to create a complete game for Windows Phone operating system. The project was carried out on Nokia Lumina 710. By the time writing this thesis, Rapid Roller was updated to version 1.1 which gained more than three thousand downloads and many good feedbacks from players.

The game name is Rapid Roller because basically player has to roll a ball rapidly to survive. It has four levels and six different balls for player to choose. When player complete one level, they will unlock a new ball. By using accelerometer, Rapid Roller makes the difference. Player can control the ball by moving the phone. The first version 1.0 was published on February.

Keywords: Windows Phone, XNA, Game

ACKNOWLEDGEMENT

I would like to thanks my mother, who taught me everything, my father who loves his family most and to all my relatives for their enormous care and support.

I am graceful to express my gratitude to my supervisor, Dr.Ghodrat Moghadampour for his help throughout my studies.

I appreciate the quality education and practical knowledge provided by VAMK.

I would like to use this opportunity to thanks all my friends for their support and encouragement during the study period.

ABBREVIATION

UI	User Interface
SDK	Software Development Kit
WP	Windows Phone
CIL	Common Intermediate Language

Table of Contents

FIGURES

1 INTRODUCTION

In this smart phone century, the definition of phone has changed. It is not only a phone but also a computer, a game console or a music player. Users can also use for surfing web, reading news, checking social, playing game, listening to music, writing note or chat with friend. Mobile phone or smart phone has become so important to human that we keep it near us all the time, even when we eat or sleep. Entertainment is one of the most attracted features of smart phone. And game is a big part of entertainment. As the result, mobile games have growth rapidly with smart phone trend.

Game development is not only limited to coding and selling. It is not so hard to make a game, but to be success is truly hard. It is not only an idea, but also the combination of attractive design, nice music, and interesting story. To make it clear, on the programming side one have to worry about input, audio, and graphics programming. Moreover, needs to think how the game is going to behave, how the design is going to be, and then translate it to the programmable language [1]. On the design side, one have to think about color, size, symbol, icon, font, music or sound effect and how their combination on real time appears. All of that need to be combined together to build up an additive and nice game.

Rapid Roller is using accelerometer on the phone for the real control of the ball as it passes through the obstacles. Score is increased by the time player survives. Unlike other games, Rapid Roller aims to use the accelerometer to control the ball movement to generate real feelings.

2 WINDOWS PHONE OPERATING SYSTEM

Being late but WP (Windows Phone) is a great mobile OS (Operating System). In other word, Microsoft built it with all the features that user expect on Android and iOS: multi-touch, social, music, email, game, office, navigation and so on. The Microsoft Office is also integrated inside WP. Especially, it has a modern user interface (UI): Metro. It is a totally new UI. Metro UI takes that idea of "King County Metro" – public transit signs. It is extremely clear and understanding. It's fast and in motion. It's about content and typography [2]. And it's entirely authentic.

Multitasking is a big feature on Android OS. Unfortunately, background programs will consume resources and kill the battery when running in the background. So balancing resources and multitasking is always a difficult issue for any mobile OS. On WP7, Microsoft invented a new way to handle multitasking: *tombstoning*. To make it clear, tombstoning is similar with hibernate-state on desktop environment but it applies for every background-app. When user change to other app, the currently app status will save necessary data to the memory – consider as "grave". The app does not running at all. It does not consume any resources. And when user comes back, it will be load from scratch. But it also uses the previous data and reloads the status – consider as "returning from grave". The tombstoning is not a real multitasking but it is an excellent replacement for multitasking. It is not only saving resources but also giving multitasking-experiences to users. [3]

Another great thing about WP OS is: Microsoft gives a standard hardware specification. It will prevent fragment and keep good quality for any WP devices.

- Set of hardware controls and buttons that include the Star, Search and Back buttons.
- A large WVGA (800x480) display screen.
- Capacity 4-point multi-touch screen.
- Support for data connectivity using cellular and Wi-Fi.

- More than 250Mb RAM and 8GB storage.
- A-GPS.
- Accelerometer. [4]

3 OVERVIEW OF TECHNOLOGIES

The official supported languages are C# and Visual Basic. Silverlight and XNA framework are used to develop application. They both base on .NET framework. Silverlight and XNA share some libraries, but not all. On the new version 7.1, programmer can use Silverlight and XNA libraries together which is not possible on version 7.0.

Silverlight is the main framework for development WP application. On the other hand, XNA is a pure game framework. There is not much different between them. Programmer can use Silverlight to develop game or use XNA to develop application. The main different is convenient. Programmer can make a simple game with Silverlight. But to make a high-performance game, XNA is the best choice. It has more graphic and animation functions. So XNA has been using to develop this game project.

3.1 C# and .NET Framework

C# is the newest object oriented programming language which has invented by Microsoft. It inherits many features of C, C++, Visual Basic, and Java. Moreover, "C# simplifies the syntax and has many advantages such as null value, enumerations, delegates, lambda expressions and direct memory access, which are not found in Java". Header file is removed. There is no requirement method and types need to be declared in order [5]. C# is very convenient for application development, especially with Microsoft Visual Studio .NET.

.NET Framework is a runtime environment which mainly focuses on Windows OS. "It consists of the common language runtime, which provides memory management and other system services, and an extensive class library, which

enables programmers to take advantage of robust, reliable code for all major areas of application development"[6]. Some of the .NET Framework services are:

- Memory management: .NET runtime will automatically take care of the memory management
- A common type system: basic types are defined by .NET Framework
- An extensive class library: programmers do not have to write code to handle low-level programming. .NET provides readily class libraries
- Development framework and technology: .NET supports many area application developments, such as ASP.NET for web-development, ADO.NET for data access, and Windows Communication Foundation for services-oriented applications
- Language interoperability: Language compilers that target the .NET Framework emit an intermediate code named Common Intermediate Language (CIL)
- Version compatibility: .NET applications are compatible to run on later version
- Multi-targeting: Project can be assembly to work on multiple platforms such as .NET Framework, Silverlight, Windows Phone, or Xbox. [7]

3.2 XNA Platform

For many years, DirectX has been graphic API for Microsoft. It is powerful and support 2D and 3D graphics effect. But the use of DirectX is just only for C and C++ language. Therefore, XNA was launched to support interface for .NET developer. It has a full managed interface for any .NET language and most of DirectX functionality [8].

The essential task of a XNA program is moving sprites (picture or text) around the screen. Sprites can be controlled by user or moving by their own schedule. Therefore, there are basic two things need to be declared: target (picture/text) and its position (Vector, Point, and Rectangle). The (0, 0) point is always set to the top

left of the phone – even the phone in landscape or portrait mod. The X-axis points to left and Y-axis points down.

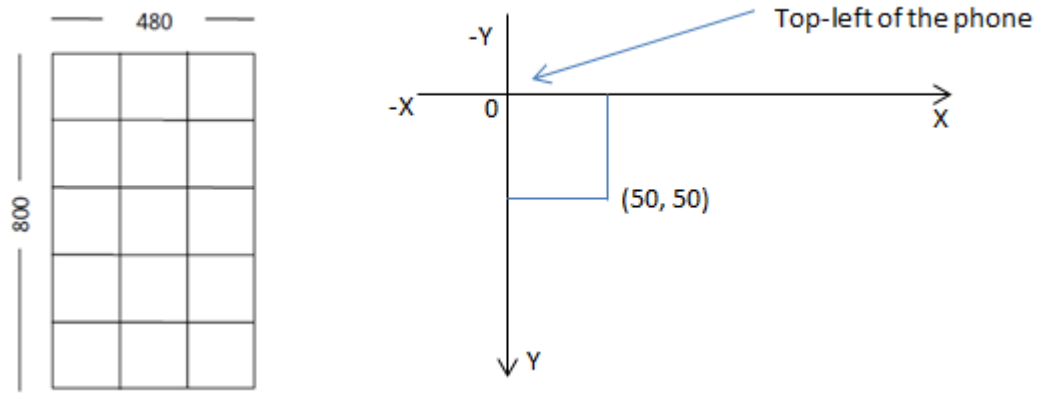


Figure 1: XNA coordinate

By default, a WP XNA project wills has two main parts: project source code (References, Prosperities, codes) and project content (Fonts, Graphics and Sound). There are two code files Program.cs and Game1.cs. The Program.cs is only used in WINDOWS or XBOX project. It is not really necessary for WP project. The Game1 class is the main class to run the project. It is inheritance to Microsoft.Xna.Framework.Game [9].

There are five methods in Game1 class: Initialize(), LoadContent(), UploadContent(), Update() and Draw().

- Initialize() : call only one time: load required services and load non-graphic content.
- LoadContent() : call only one time in the game: load graphic content
- UploadContent() : call only one time: to upload all content
- Update() : is called 30 times in one second
- Draw() : draw sprites, called after the Update()

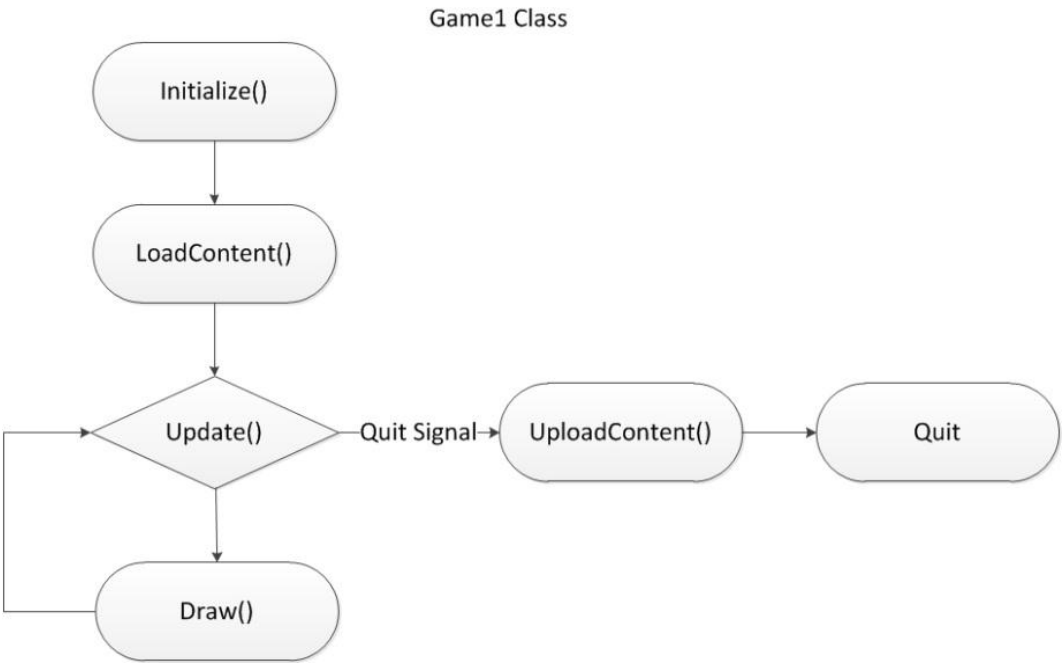


Figure 2: Game1 class

3.3 Windows Phone Software Development Toolkit (SDK)

Microsoft offers the Windows Phone SDK for free. It includes:

- Microsoft Visual Studio 2010 Express for Windows Phone.
- Windows Phone Emulator.
- Windows Phone SDK 7.1 Assemblies.
- Silverlight 4 SDK and DRT.
- Windows Phone SDK 7.1 Extensions for XNA Game Studio 4.0.
- Microsoft Expression Blend SDK for Windows Phone 7.
- Microsoft Expression Blend SDK for Windows Phone OS 7.1.
- WCF Data Services Client for Window Phone.
- Microsoft Advertising SDK for Windows Phone. [10]

When programmer debugs WP project, Visual Studio Express IDE will automatically connect with Windows Phone Emulator. Programmer can test application on emulator before deploy on the phone.

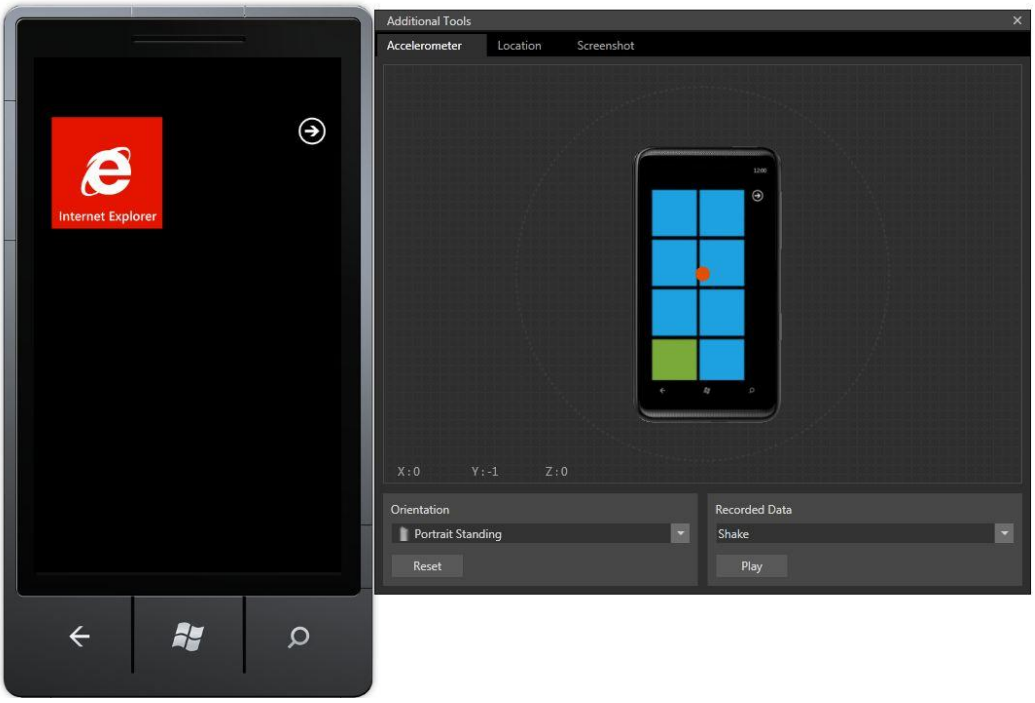


Figure 3: Windows Phone Emulator

4 RAPID ROLLER

4.1 Rapid Roller Main Functions

Rapid Roller is arcade game. In Rapid Roller, there is a ball fall down and obstacles (bars) are moving up. Player has to control the ball pass through these obstacles. Game will end when the ball hit the top panel. Over time, score will be increased.

Main functions:

- Move the ball by moving the phone.
- View tutorial.
- Choose level.
- View high score.
- Change ball.
- Turn on/off sound.
- Quit game.

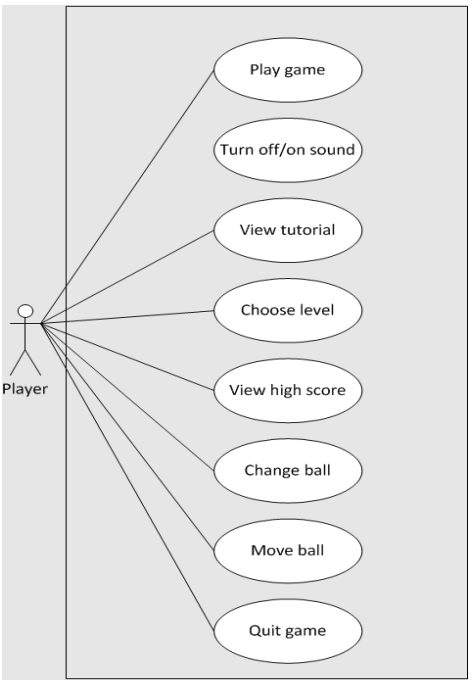


Figure 4: Rapid Roller use case diagram

Rapid Roller has sevens classes. Every class updates and displays different component of the game. MainGameScreen class is the core. It will call other classes.

Most of classes will have three main function: LoadContent(), Update(), and Draw(). LoadContent is responsibility for initializing necessary content (variables, load images or music). Update() is responsibility for handle the class function. Most likely Update() will control the position or trigger of displaying the image. Finally Draw() function is responsibility for displaying image to the phone screen.

MainGameScreen has most of important functions of the game:

- CheckMusicCanPlay(): Check player is listening to music or not.
- Restart(): restart the game.
- OpenSaveFile(): Open and get the score in file.
- SaveHighScore(): Save high score to file.
- LoopMusic(): Looping music background.
- PlaySoundEffect(): Play sound effect.
- BarGenerate(): generate bars by time.
- BarUpdate(): handle bars update and position. Remove the bar when it goes out of the screen.
- AccelerometerReadingChanged(): Reading accelerometer changes
- NewReading(): convert accelerometer changes into integer.
- CollisionDetect(): Detect collision of the ball and bars.
- HandInput(): Handle the input touch from player.

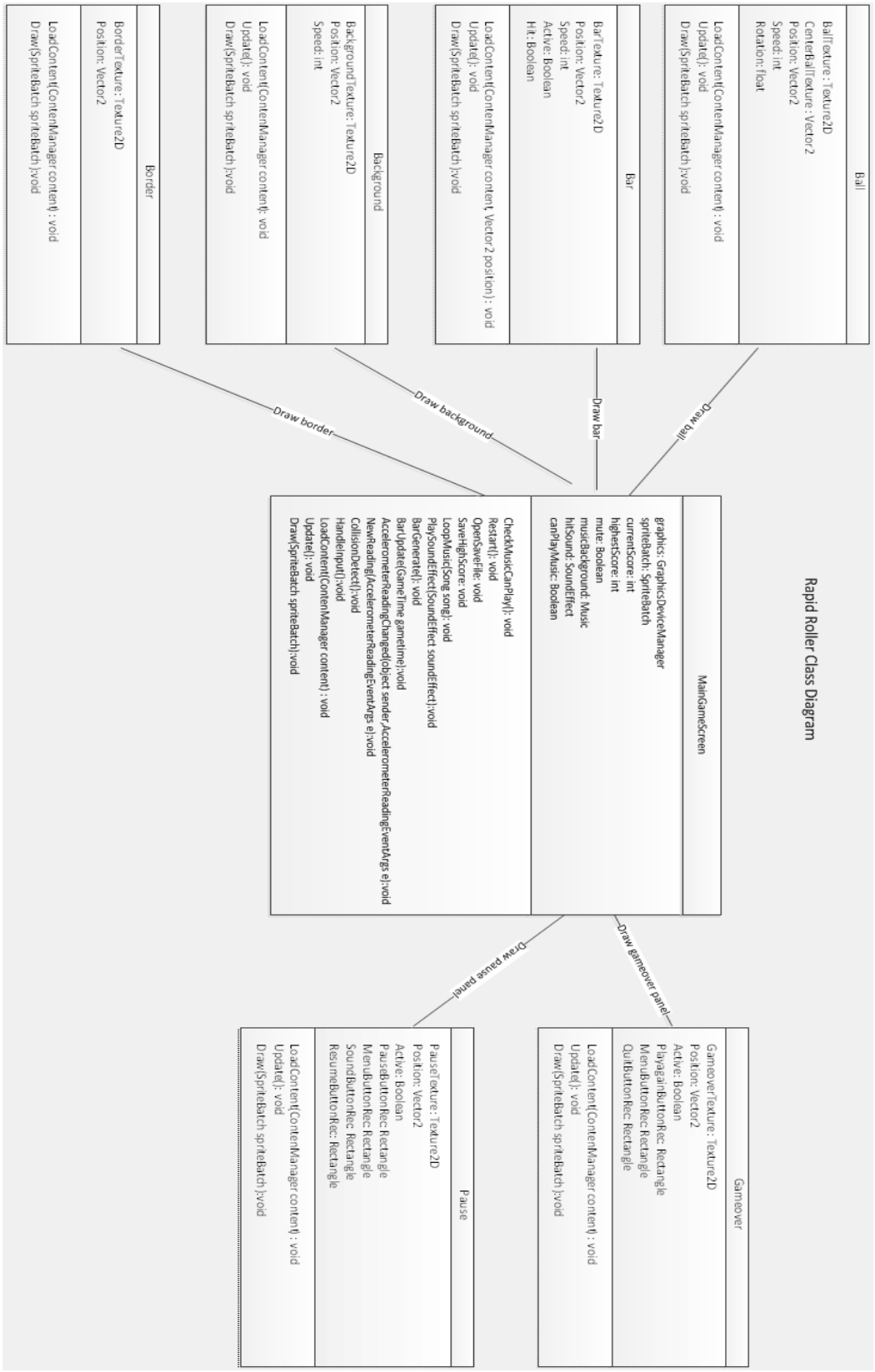


Figure 5: Rapid Roller class diagram

Accelerometer is main part of controling the ball movement. When player move the phone, accelerometer will send and a horizontal value – left or right. Then the ball position will be update and draw again.

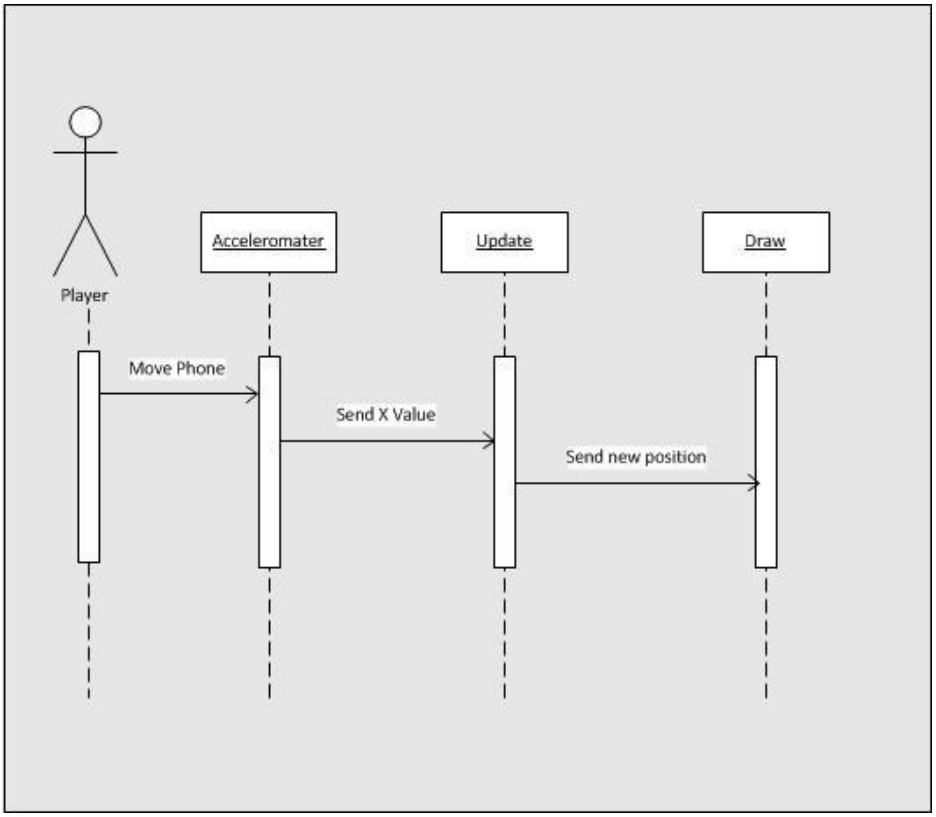


Figure 6: Ball-control sequence diagram

Pause game button is resposibility to detect the press from player. When player press the pause button, pause signal will be send. Then it will stop updating the other images position and start to move the pause panel to the screen.

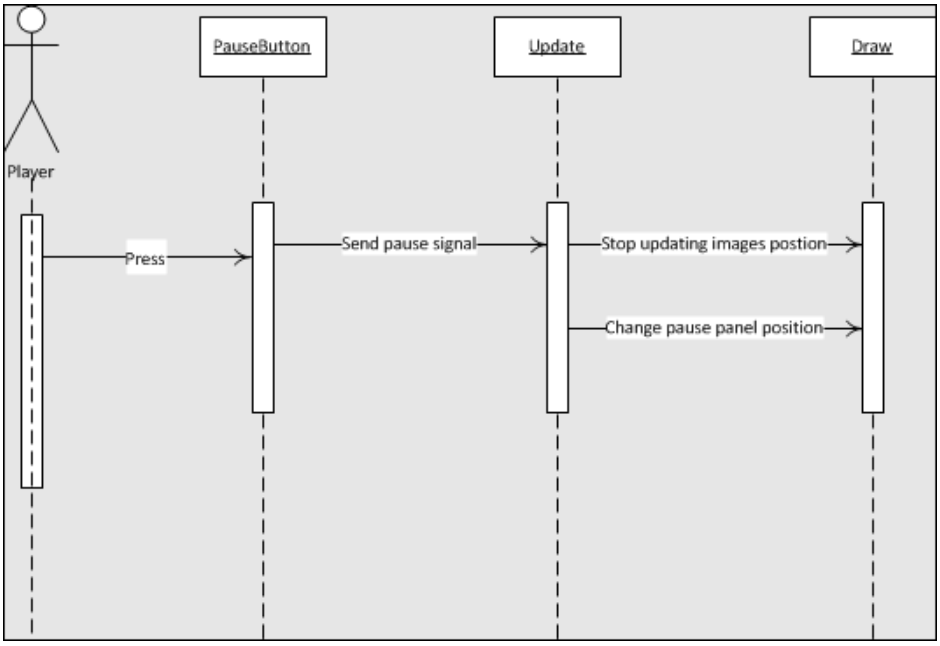


Figure 7: Pause game sequence diagram

4.2 Rapid Roller Design

Designing state is a difficult part for programmer, because it is clearly not mathematic or coding. It is art. Programmer has to think and explain the game idea to the designer. So designer could understand and create the graphics which are suitable, nice and attractive for the game. Designer and programmer need to understand other ideas clearly and work together most of the time. From the beginning, the color, the backgrounds, shapes and layout will be decided first. Then designer make some sample and send to programmer. Programmer puts it into the game, tests it and sees how it looks on the real phone. If there is something needs to be change – most likely there is – programmer will report to designer. Then designer has to draw it again. For example, when develop Rapid Roller, to find out the ideally ball, I have to test the ball size and color on the real phone many time. If it went wrong, designer has to draw it again for me.

General of Rapid Roller design is wood and black color. The following table contains design information. Measurement is pixel. I implemented Rapid Roller by following layouts order. The next layout will cover previous layout. Base on the design and function of graphics, Rapid Roller is divided into 5 layouts:

- Layout 1: Background scrolling up
- Layout 2: Bars
- Layout 3: Border
- Layout 4: Ball
- Layout 5: Notification game over, pause panel, pause button and score

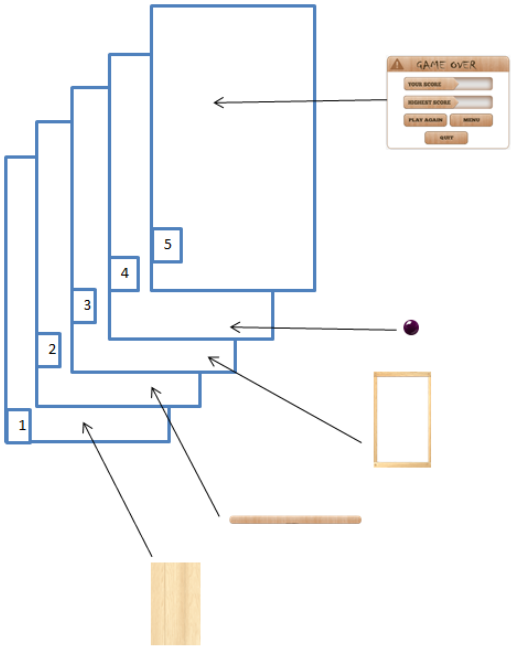


Figure 8: Rapid Roller layouts

The flowing picture is how the game look likes after draw all the layouts.

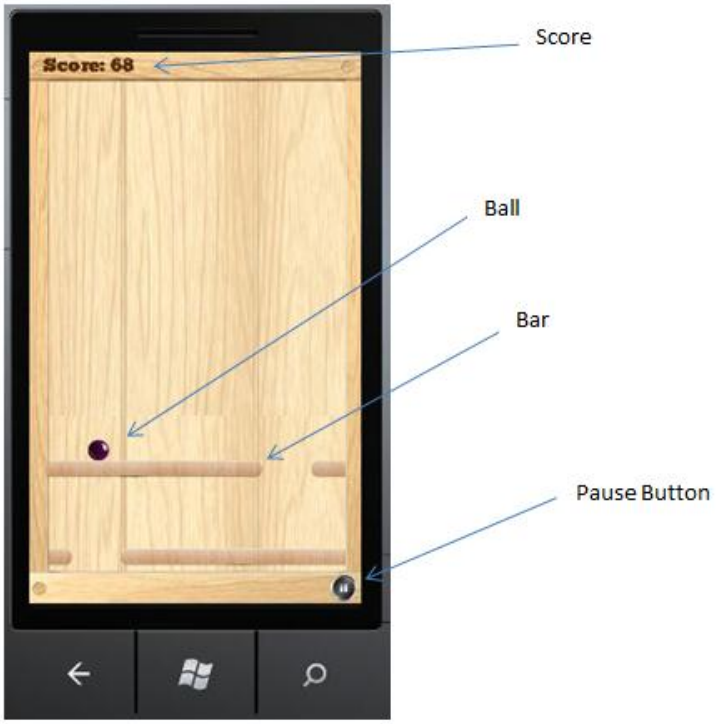


Figure 9: Rapid Roller prototype

5 RAPID ROLLER IMPLEMENTATION

5.1 First Layout: Background

In order to make player feel like the ball drop down, background will be moved up all the time. Player's eyes will have vision that ball drop down very fast.

Background class will has three methods:

- LoadContent(): Load the necessary graphics content
- Update(): Update the background position
- Draw(): Draw background picture to the screen.

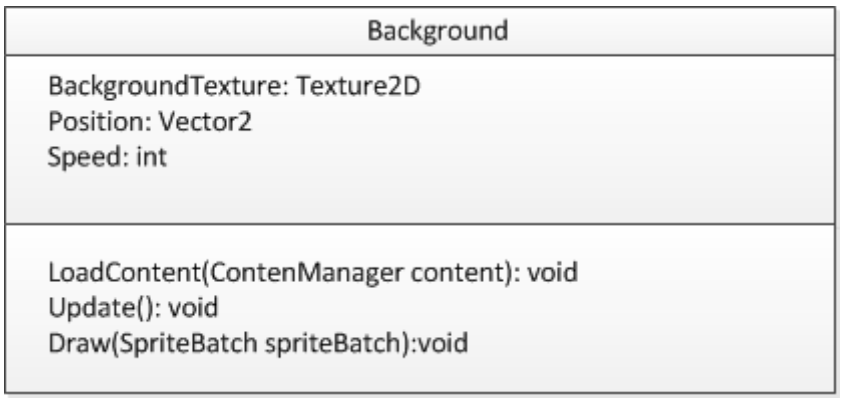


Figure 10: Background class

Background position is a vector. It will be minus 4 pixels every Update(). It will look like background moving up constantly. However, there is gap when background picture move out of it initial position.

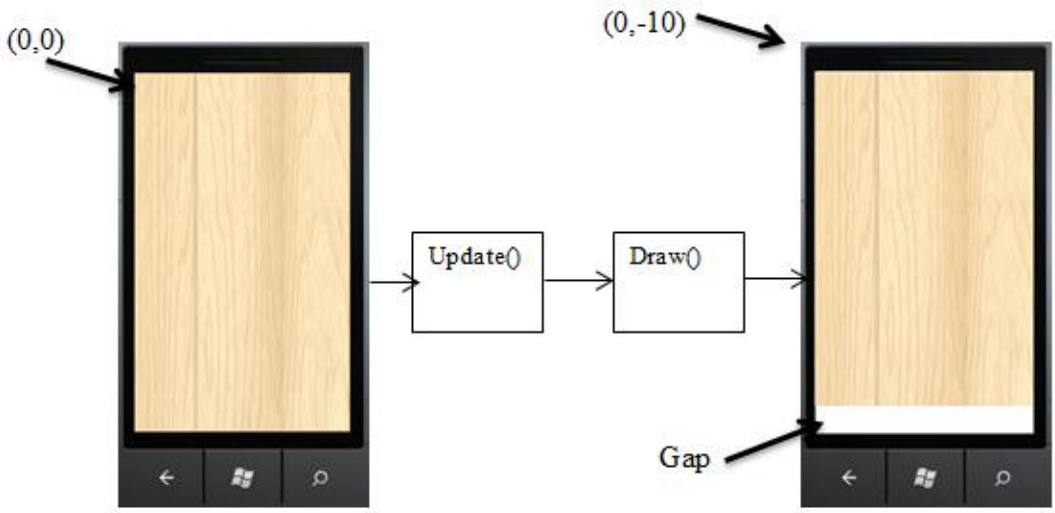


Figure 11: Background scrolling gap

Designer could draw background's height longer, but it not possible because we do not know how long it needs to be. It depends on how long player play. And background picture gets bigger also mean the game is going to be heavier.

In order to make the background scrolling up infinitive, the Draw() method will draw two background pictures at the same time and next to each other. When the first picture moves up, the next one also move up and cover the gap. When the first picture move out of the screen (the second picture is on the screen), both pictures will be set back to initial position.

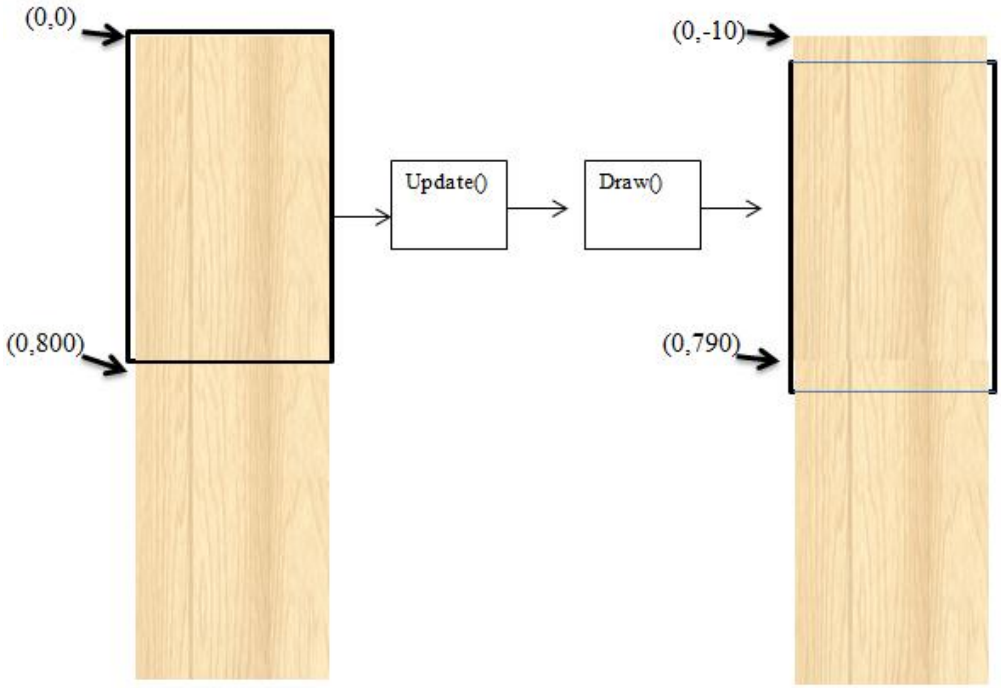


Figure 12: Background scrolling infinitive

The follwing code is the background-class code. It has reposibility to scrolling background image infinite.

```
Texture2D backgroundTexture;
        Vector2 Position1;
        Vector2 Position2;
        public int Speed;

        public void LoadContent(ContentManager content)
        {
            //Load background image to backgroundTexture
            backgroundTexture = content.Load<Texture2D>("Background");
            //Speed is 4 pixel
            Speed = -4;
            Position1 = new Vector2(0, 0);
            Position2 = new Vector2(0, 800);
        }

        public void Update()
        {
            // Both 2 backgrounds goes up at the same speed(4 pixel)
            Position1.Y += Speed;
            Position2.Y += Speed;

            // if first picture goes out of the screen, draw both again
            if (Position1.Y <= -800)
            {
                Position1.Y = 0;
```

```
                Position2.Y = 800;
            }
        }

        public void Draw(SpriteBatch spriteBatch)
        {
            // Draw first background
            spriteBatch.Draw(backgroundTexture, Position1, Color.White);
            // Draw second background
            spriteBatch.Draw(backgroundTexture, Position2, Color.White);
        }
    }
```

5.2 Second Layout: Bars

Bar is obstacle. It moves from bottom to the top and have a small gap for the ball to pass. The gap needs to have constant size – just enough - for the ball get through. When the bar goes out the screen, it will be removed.

If the bar goes out of the screen, its "Active" variable will be set to false. Then it will be removed. We also need a "Hit" variable to check the collision between the ball and bar. Bar class has three methods:

- LoadContent(): Load the necessary graphics content
- Update(): Update the bar status and position
- Draw(): Draw bar picture to the screen.

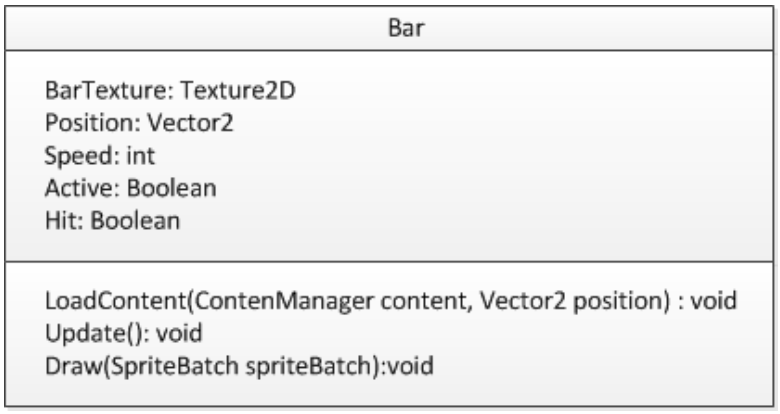


Figure 13: Bar class

To manage collision, bar will be generated and add to a list in main game. The gap will be created by two bars at the same time. They will have the same Y position. First X position is random. But the second bar X position is take by first bar X plus the bar width and the gap size. Then it is easy to handle collision later.

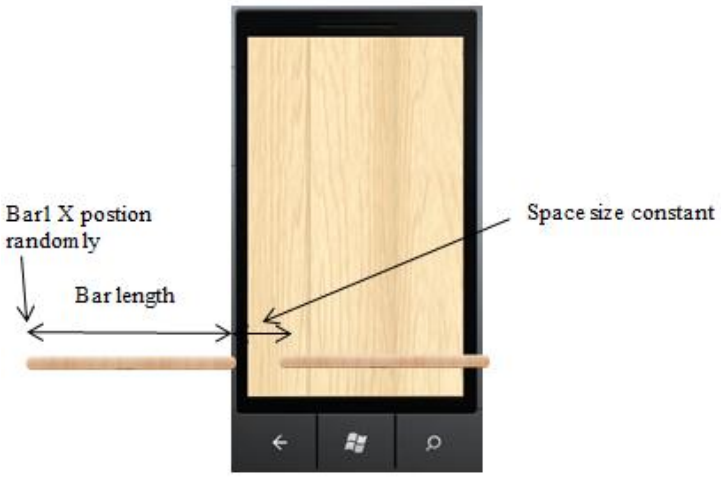


Figure 14: Bars location

The following code is bar-class code. It will resposiblity to draw the bar image to the screen.

```
class Bar
    {
        public Texture2D BarTexture;
        public Vector2 Position;
        public float Speed;
        public bool Active;
        public Boolean Hit;
        public void LoadContent(ContentManager content, Vector2
position)
        {
            // Load bar image to BarTexture
            BarTexture = content.Load<Texture2D>("Bar");
            // Initialize bar position
            Position = position;
            // set bar's speed to 4 pixel (moving up)
            Speed = -4;
            Hit = false;
            Active = true;
        }
        public void Update()
        {
            // update bar position
            Position.Y += Speed;
        }
        public void Draw(SpriteBatch spriteBatch)
        {
```

```
            // draw bar texture to the screen
            spriteBatch.Draw(BarTexture, Position, null, Color.White, 0,
Vector2.Zero, 1f, SpriteEffects.None, 0f);
        }
    }
```

The following code is bar-generation. It will automatically, create bar with random position. Then add the bar to the bar-list.

```
#region Bar Generator and Update
        // list to keep bars
        List<Bar> _bars = new List<Bar>();
        // set previousTimeSpan to zero
        TimeSpan _previousTimeSpan = TimeSpan.Zero;
        // set nextTimeSpan to 2 second
        TimeSpan _nextTimeSpan = TimeSpan.FromSeconds(2f);
        Random _random;
        private void BarGenerate()
        {
            Bar _bar1 = new Bar();
            Bar _bar2 = new Bar();
            _random = new Random();
            // first bar position is randomly
            Vector2 _barPostiion1 = new Vector2(_random.Next(-400,
GraphicsDevice.Viewport.Width - 400), 810);
            // second bar position is at the same line, but separate
with first bar by 430 pixel
            // This create a constantly space between 2 bars which the
ball can go through
            Vector2 _barPostiion2 = new Vector2(_barPostiion1.X + 430,
810);
            _bar1.LoadContent(Content, _barPostiion1);
            _bar2.LoadContent(Content, _barPostiion2);
            // add 2 bar to bars list
            _bars.Add(_bar1);
            _bars.Add(_bar2);
        }
        private void BarUpdate(GameTime _gametime)
        {

            // regenerte bar every 2s
            if (_gametime.TotalGameTime - _previousTimeSpan >=
_nextTimeSpan)
            {
                _previousTimeSpan = _gametime.TotalGameTime;
                BarGenerate();
            }

            // when bar goes out of the screen , remove it
            for (int i = 0; i < _bars.Count(); i++)
            {
                if (_bars[i].Position.Y < -50)
                    _bars.RemoveAt(i);
                _bars[i].Update();
```

```
            }
        }
#endregion
```

5.3 Third Layout: Border

Border is placed over background and bars. It also reserves the space for score and pause button. Rapid Roller will look clearer.

Border class is simple. It does not need Update() function. It just needs to draw the border picture on the screen. The important is the border picture need to have transparent space inside. Then player could see background and bars.

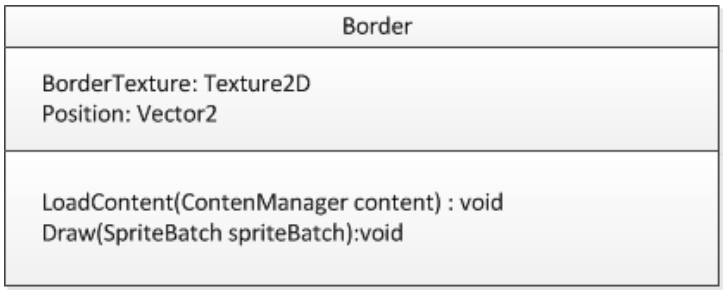


Figure 15: Border class

Following figure show how the border will be displayed on the emulator.

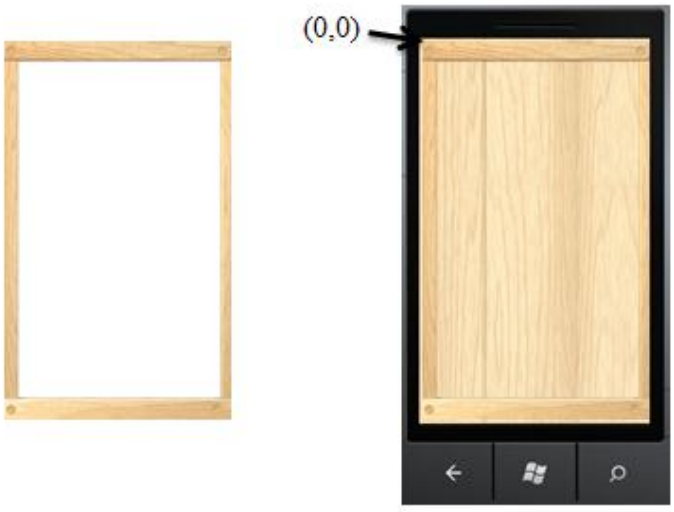


Figure 16: Border

The following is the border-class code. It just display the border to the screen

```
class Border
    {
        Texture2D borderTexture;
        Vector2 borderPosition;
        public void LoadContent(ContentManager content)
        {
            // Load broder image to the barTexture
            borderTexture = content.Load<Texture2D>("Border");
            // set border position
            borderPosition = new Vector2(0, 0);
        }
        public void Draw(SpriteBatch spriteBatch)
        {
            // draw border
            spriteBatch.Draw(borderTexture, borderPosition,
Color.White);
        }
    }
```

5.4 Fourth Layout: Ball

Ball class will have gravity simulation. Ball picture will drop down when it is not on any bar. It creates vision that the ball fall down by gravity. When ball hit on the bar, it will stop falling down and goes up with the bar speed. We will handle the collision and accelerometer moment in the main game class.

In order to make the ball looks nicer, ball is rotated to left when it goes left and to right when it goes right. Therefore, in ball class we need a “Rotation” variable. It defines the angle rotation for the ball picture. By default, XNA draws a picture with coordinate on the top-left. In order to scroll the ball center, we need to provide a vector which is point to the center of the ball picture.

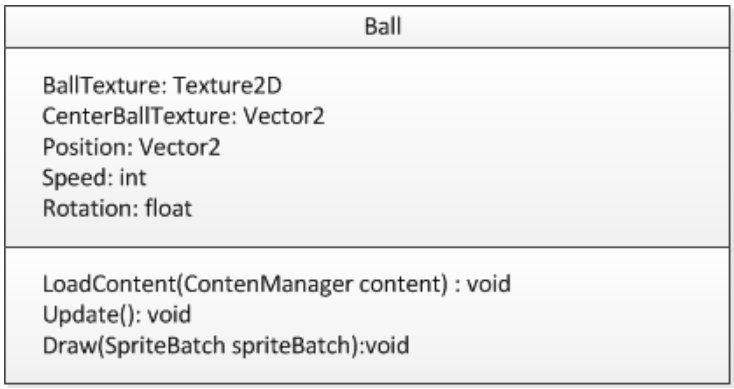


Figure 17: Ball class

The following figure present how the ball will be displayed on the emulator.

Figure 18: Ball falling down

The following code is ball-class code. It is responsible to draw the ball drop down by the gravity.

```
class Ball
    {
        public Texture2D BallTexture;
        public Vector2 Position;
        public float Gavity = 6;
        public float Speed;
        public float Rotation;
        Vector2 BallCenter;
        public void LoadContent(ContentManager content)
        {
            // load ball image to BallTexture
            BallTexture = content.Load<Texture2D>("Ball");
```

```
            Rotation = 0f;
            // set the first position for the ball
            Position = new Vector2(100,50);
            BallCenter.X = BallTexture.Width / 2;
            BallCenter.Y = BallTexture.Height / 2;
            // set ball seep to gravity (6)
            Speed = Gavity;
        }
        public void Update()
        {
            // update ball position through speed
            Position.Y += Speed;
            // keep the ball inside border
            this.Position.X = MathHelper.Clamp(this.Position.X, 45, 480
- BallTexture.Width -10);
            this.Position.Y = MathHelper.Clamp(this.Position.Y, 45, 800
- BallTexture.Height - 30);
        }

        public void Draw(SpriteBatch spriteBatch)
        {
            // draw ball
            spriteBatch.Draw(BallTexture, Position, null, Color.White,
Rotation, BallCenter, 1f, SpriteEffects.None, 0f);
        }
    }
```

5.5 Fifth Layout: Score, Pause and Gameover

Score is calculated by time player survives. It will be draw by main game class, we just add one score every Update() method is called. Score font is Comic San. It is placed on the top left of the game border. So it is clear and easy for player to follow the score.

Pause class has two main functions: draw the pause button and draw up the pause panel when pause button is clicked. To detect the input touch with button, pause class need to declare the rectangles which cover buttons.

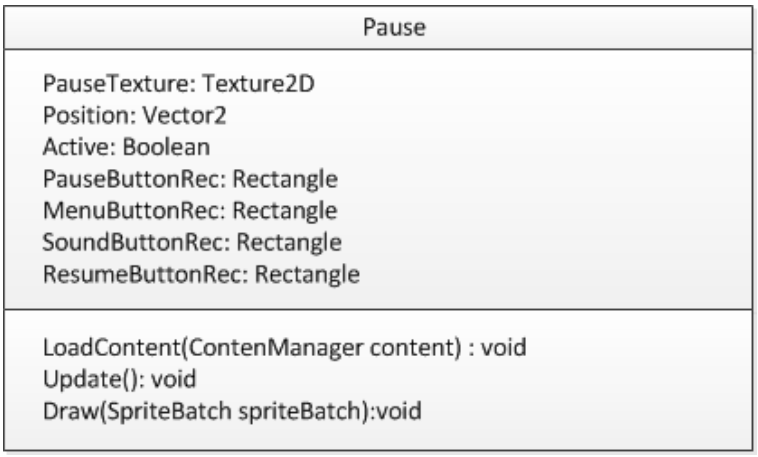


Figure 19: Pause class

Game-over class draws the notification game over. It will be active when the ball hit top, then the game stops and game-over panel slowly moving down. Game-over class also need the rectangles cover its buttons.

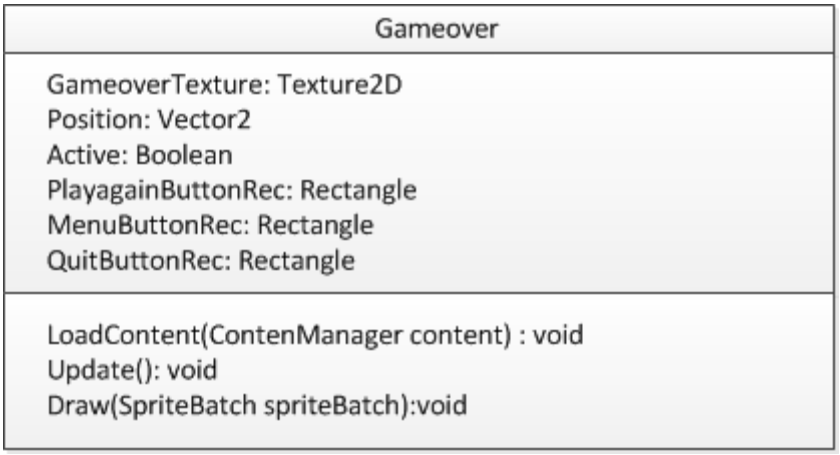


Figure 20: Game-over class

The following figure is how the the game-over panel and pause panel will be display when activate

Figure 21: Score, pause and gameover

The following code is pause-class code. It responsible to draw pause-panel to the screen.

```
class Pause
    {
        Texture2D pauseTexture;
        Vector2 pausePosition;
        Texture2D panelTexture;
        Vector2 panelPosition;
        public Boolean active;
        public Rectangle RecMenu = new Rectangle(10,761,111,35);
        public Rectangle RecSound = new Rectangle(135, 761, 111, 35);
        public Rectangle RecResume = new Rectangle(269, 761, 120, 35);
```

```
        public Rectangle RecPauseButton() {
            // rectangle cover pause button
            return new
Rectangle((int)pausePosition.X,(int)pausePosition.Y,pauseTexture.Width,p
auseTexture.Height);
        }

        public void LoadContent(ContentManager content)
        {
            // load pause button image to pauseTexture
            pauseTexture = content.Load<Texture2D>("PauseImage");
            // load pause panel inmage to panelTexture
            panelTexture = content.Load<Texture2D>("PausePanel");
            pausePosition = new Vector2(435, 757);
            panelPosition = new Vector2(0, 810);
            active = false;
        }

        public void Update()
        {
            // pause panel moving up
            if (this.active)
            {
                if (panelPosition.Y >= 760)
                    panelPosition.Y -= 5;
            }
            else
                // pause panel moving down
                if (panelPosition.Y <= 810)
                    panelPosition.Y += 5;
        }

        public void Draw(SpriteBatch spriteBatch)
        {
            // draw pause button
            spriteBatch.Draw(pauseTexture, pausePosition, Color.White);
            // draw pause panel
            spriteBatch.Draw(panelTexture, panelPosition, Color.White);
        }
    }
```

The following is game-over class code. It responsible to display the game-over panel to the screen.

```
class Gameover
    {
        Texture2D gameoverPanel;
        public Vector2 position;
        public Boolean active;
        public Rectangle RecAgain = new Rectangle(129, 428, 108, 33);
        public Rectangle RecMenu = new Rectangle(243, 428, 108, 33);
        public Rectangle RecQuit = new Rectangle(182, 472, 108, 33);

        public void LoadContent(ContentManager content)
```

```
        {
            // load game over panel to gameoverTexture
            gameoverPanel = content.Load<Texture2D>("GameOver");
            position = new Vector2(240, -300);
            active = false;
        }
        public void Update()
        {
            // gameover panel moving down
            if (active)
            {
                if (position.Y < 400)
                    position.Y += 20;
            }

        }
        public void Draw(SpriteBatch spriteBatch)
        {
            // draw game over panel
            if (active)
                spriteBatch.Draw(gameoverPanel, position, null,
Color.White, 0, new Vector2(gameoverPanel.Width / 2,
gameoverPanel.Height / 2), 1f, SpriteEffects.None, 0);
        }
    }
```

5.6 Ball control

Every Windows phone has an accelerometer to measure the force. When the phone moves, accelerometer will response with gravity of the Earth. The accelerometer output is three dimensions vector (x,y,z). [11]

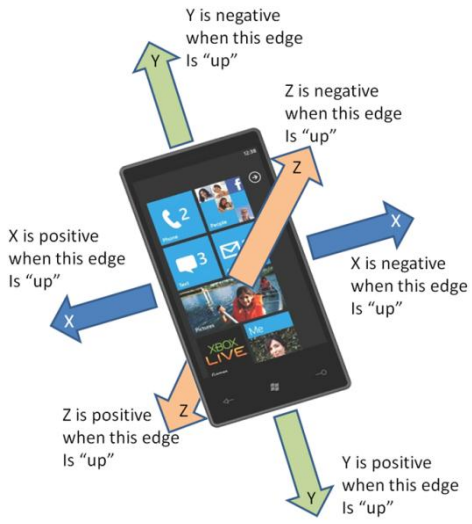


Figure 22: Accelerometer output vectors [12]

Only x value from accelerometer is needed to detect ball scrolling left or right. When accelerometer x is bigger than 0, it means that the phone lift to right. Ball location x will be increased. And when accelerometer x is smaller than 0, ball location x will be decreased.

Figure 23: Ball scrolling

The following code is how the accelerometer is called and invoke.

```
#region Accelerometer Fucntion

        Accelerometer accelerometer = new Accelerometer();
        double X;
        void AccelerometerReadingChanged(object sender,
AccelerometerReadingEventArgs e)
        {
            Deployment.Current.Dispatcher.BeginInvoke(() =>
NewReading(e));
        }
        void NewReading(AccelerometerReadingEventArgs e)
        {
            X = e.X;
        }
#endregion
```

The following code is how the ball position will be change through the accelerometer horizontal value – x value.

```
#region Acceleremeter update ball position
                    if (X < 0)
                    {
                        // rotate the ball picture to left
                        _ball.Rotation -= 0.3f;
                        // move the ball position to left
                        _ball.Position.X += (float)X * 12;

                    }
                    else
                    {
                        // rotate the ball picture to right
                        _ball.Rotation += 0.3f;
                        // move the ball position to right
                        _ball.Position.X += (float)X * 12;
                    }
#endregion
```

5.7 Detect Collision

There is collision between the ball and bars. When the ball hit on the bar, it stops drowning down and is carried up by the bar. In other word, we will change the ball speed to be the same with the bar. Player will see vision that the ball is blocked by the bar.

To detect the collision, we generate invisible rectangles cover around the ball and bars. And on every Update(), collision is found by checking is there any bars' rectangle contain ball's rectangle. If there is, the speed is changed to bar speed.

Figure 24: Detect collision

The following code is detect the collision function. We also have to check the bar position. If it is upper the ball position. The we do not need to check its rectangle.

```
private void CollisionDetect()
        {
            Rectangle rectangle1;
            Rectangle rectangle2;
            // rectangle1 cover the ball
            rectangle1 = new Rectangle((int)_ball.Position.X,
(int)_ball.Position.Y, _ball.BallTexture.Width,
_ball.BallTexture.Height);

            for (int i = 0; i < _bars.Count; i++)
            {
                // rectangle2 cover bars
                rectangle2 = new Rectangle((int)_bars[i].Position.X +
20, (int)_bars[i].Position.Y, _bars[i].BarTexture.Width - 8,
_bars[i].BarTexture.Height);
                if (_bars[i].Active)
                {
                    if (!_bars[i].Hit)
                    {
                        // if the ball position is
                        // smaller bar position
                        // do not check becasue the
                        // ball already pass through the bar
                        if (_ball.Position.Y <= _bars[i].Position.Y)
                            // if rectangle1 intersect rectangle2
                            // => ball hit bar
                            if (rectangle1.Intersects(rectangle2))
                            {
                                // play hit sound
                                PlaySoundEffect(_hitSound);
                                _bars[i].Hit = true;
                                // ball speed take the bar speed value.
                                // so the ball moving up
                                _ball.Speed = _bars[i].Speed;
                                // keep the ball texture
                                // not to step on bar texture
                                _ball.Position.Y =
MathHelper.Clamp(_ball.Position.Y, _bars[i].Position.Y - 18,
_bars[i].Position.Y - 16);
                            }
                    }
                    else
                    {
                        if (!rectangle1.Intersects(rectangle2))
                        {
                            // do not check this bar anymore
                            _bars[i].Active = false;
                            // turn ball speed to gavity
                            //=> ball continue drown down
                            _ball.Speed = _ball.Gavity;
                        }
                    }

                }
            }
        }
```

5.8 Handle Input Touch

WP7 touch screen requirement is at least four fingers simultaneous. In Silverlight program, touch input is obtained through event. In XNA, touch is handle by static class called during Update() method. [13]

There are two types touch in XNA: low- level touch and gestures touch

- Low-level touch: Pressed, Move and Released.
- Gestures touch: Tap, Double Tap, Flick, Hold, Pinch, Pinch Complete, Free Drag, Horizontal Drag, Vertical Drag, and Drag Complete.

There is pause button and menu bar need the player input touch. It is similar with collision detect. We also generate invisible rectangles that covers the menu buttons and when the player touches the screen, touch point will be recorded. If a button's rectangle contains that touch point, it function will be active.

Figure 25: Rectangles cover button

The following code is handle input touch function. It will detech wherever player touch on the right button and trigger the signal to be true.

```
private void HandleInput()
        {
            TouchCollection _touches = TouchPanel.GetState();
            if (_touches.Count > 0 && _touches[0].State ==
TouchLocationState.Pressed)
            {
                Microsoft.Xna.Framework.Point _touchPoint = new
Microsoft.Xna.Framework.Point((int)_touches[0].Position.X,
(int)_touches[0].Position.Y);
                if (!_gameover.active)
                {
                    #region Pause Panel Input
                    // if player touch pause button
                    if (_pause.RecPauseButton().Contains(_touchPoint))
                    {
                        if (!_pause.active)
                            _pause.active = true;
                        else
                            _pause.active = false;
                    }
                    if (_pause.active)
                    {
                        // if player touch menu button
                        if (_pause.RecMenu.Contains(_touchPoint))
                        {
                            accelerometer.Stop();
                            this.Exit();
                        }
                        // if player touch sound button
                        if (_pause.RecSound.Contains(_touchPoint))
                        {
                            // pause and resume mmusic and sound
                            if (!_mute)
                            {
                                _mute = true;
                                MediaPlayer.Pause();
                            }
                            else
                            {
                                _mute = false;
                                MediaPlayer.Resume();
                            }
                        }
                        // if player touch resume button
                        if (_pause.RecResume.Contains(_touchPoint))
                        {
                            _pause.active = false;
                        }
                    }
                    #endregion
                }
                    else
```

```
            {
                #region Gameover Panel Input
                // if player touch play again button
                if (_gameover.RecAgain.Contains(_touchPoint))
                {
                    SaveHighScore();
                    Restart();
                }
                // if player touch menu button
                if (_gameover.RecMenu.Contains(_touchPoint))
                {
                    SaveHighScore();
                    accelerometer.Stop();
                    this.Exit();
                }
                // if player touch quit button
                if (_gameover.RecQuit.Contains(_touchPoint))
                {
                    SaveHighScore();
                    accelerometer.Stop();
                    this.Exit();
                }
                #endregion
            }
        }
    }
    #endregion
```

5.9 Music and Sound Effect

Music and sound effect are important in a game. It makes the game more attractive and excited to play. It affects the player feeling. Music background is repeatedly playing when the game starts. There are two sound effects in Rapid Roller: The ball hit bar sound and button clicked.

One important thing, when player are listening to music and want to play the game, it is compulsory to ask that do he/she want to stop listening to music. If he/she does not want to stop the music, the game has to be in silent mode. This is a requirement to publish any application to WP market.

The following code is checking is player is listening to music or not. If he/she is listening to music, then we have to send a notification which asks for permission.

```
private void CheckMusicCanPlay()
        {
            // if player does not listening to music
            // => the game has conctrol of mediaplayer
            if (MediaPlayer.GameHasControl)
            {
                _canPlayMusic = true;
                LoopMusic(_musicBackground); // play music background
            }
            else
            {
                if (MessageBox.Show("Do u want to stop listening to
music?", "You are listening to Music", MessageBoxButton.OKCancel) ==
MessageBoxResult.OK)
                {
                    // if player click ok.
                    //stop current music and play game music
                    MediaPlayer.Stop();
                    _canPlayMusic = true;
                    // play music background
                    LoopMusic(_musicBackground);
                }
            }
        }
```

The following code is reapltedly playing the background music. We just need to call it one time the LoadContent().

```
private void LoopMusic(Song song)
        {
            if (_canPlayMusic)
                if (!_mute)
                {
                    MediaPlayer.Play(song);
                    MediaPlayer.IsRepeating = true;
                }
        }
```

The following code is play sound effect. We use this function to play the ball collision sound.

```
private void PlaySoundEffect(SoundEffect soundEffect)
        {
            if (_canPlayMusic)
                if (!_mute)
                    soundEffect.Play();
        }
```

5.10 High Score

All applications on WP has its own storage called isolated storage. Different applications have different isolated storage which is not accessible from outside. [14]

When game starts, we will check is there high score file already present or not. If it does not exist, we create it. If it exists, we open and check the current highest score and if player gains bigger score, it will replace the current high score in the saved file. The following is the save file function.

```
private void OpenSaveFile()
        {
            // Get the place to store data
            using (IsolatedStorageFile isf =
                IsolatedStorageFile.GetUserStoreForApplication())
            {
                if (!isf.FileExists("HighScore.dat"))
                {
                    // If the file does not exist,
                    //create it to save the highscore data
                    using (IsolatedStorageFileStream isfs =
                        isf.CreateFile("HighScore.dat"))
                    {
                        using (StreamWriter writer = new
StreamWriter(isfs))
                        {
                            writer.WriteLine(_highestScore.ToString());
                            writer.Flush();
                            writer.Close();
                        }
                    }
                }
                else
                {
                    //if the file exist,
                    //open file and save so _highestScore
                    using (IsolatedStorageFileStream isfs =
                            isf.OpenFile("HighScore.dat",
FileMode.Open))
                    {
                        // Get the stream to read the data
                        using (StreamReader reader = new
StreamReader(isfs))
                        {
                            // Read the highscores
                            while (!reader.EndOfStream)
                            {
                                _highestScore =
int.Parse(reader.ReadLine());
                            }
```

```
                        reader.Close();
                    }
                }
            }
        }
    }
```

The following is save high score function. It will be called when player fail. It responsible to update the highest score for Rapid Roller.

```
private void SaveHighScore()
    {
        if (_currentScore > _highestScore)
        {

            using (IsolatedStorageFile isf =
IsolatedStorageFile.GetUserStoreForApplication())
            {
                using (IsolatedStorageFileStream isfs =
isf.OpenFile("HighScore.dat", FileMode.Open))
                {
                    using (StreamWriter writer = new
StreamWriter(isfs))
                    {
                        writer.WriteLine(_currentScore.ToString());
                        writer.Flush();
                        writer.Close();
                    }
                }
            }
        }
    }
```

6 TESTING

Testing has been done by using emulator and Nokia Lumina 710 with WP version 7.1. After completing every class or functions, testing was carried out immediately.

One important bug was the ball and bar collision. The ball picture sometime step over the bar or make stick inside the bar picture. In order to prevent that bug, we have to used MathHelper.Clamp() function to keep the ball position on a safe range with the bar position.

Figure 26: Ball step on bar error

According to player feedback, there is crashing bug on the version 1.0. It was because of accelerometer closes when player quits the game. So sometime, when player plays Rapid Roller again, accelerometer starts again, and then it crashes. The problem has been fixed on version 1.1, by closing the accelerometer function whenever player quit the game.

7 PUBLISH TO MARKET

After testing, before uploading to the market, application must follow the following requirements:

- Application is reliable.
- Application makes efficient use of resources.
- Application does not interfere with the phone functionality.
- Application is free of malicious. [15]

One thing to notice, unlike Silverlight, by default, XNA does not handle the back button. So the application needs to handle the back button. And like we said before, if the application is going to play music, it has to ask the permission from user.

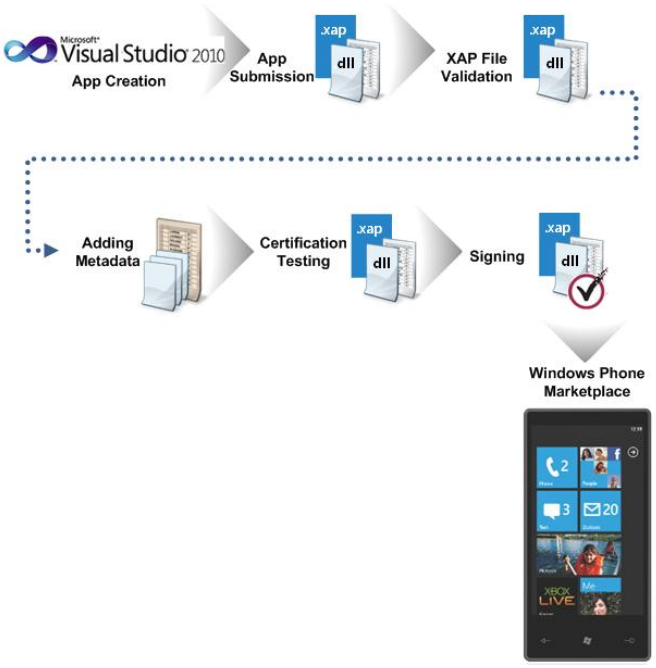


Figure 27: WP certification process [16]

8 CONCLUSION

Building a complete game is interesting. On the one hand, it requires programming and designing skills; on the other hand, the author should understand customer needs. Designing a game is not only just to program but also to think and imagine what kind of game that customers desire.

From the idea of controlling the ball with accelerometer, Rapid Roller turns into a nice game with attractive design and exciting music. It also has different levels to challenge the players. Currently, Rapid Roller has four levels and six different balls to unlock. In the future, it can be extended to more levels, more balls, and multiplayer.

The project has achieved the goal: a new valuable game to play on Windows Phone. In the first week after publishing, it has got more than one thousand download and continues to increase more. The updated version published is also successful. There is no bug on the game anymore. It can be used as the backbone to develop more games in the future. Game animation, design, music and coding skills play an important role to complete the game.

The most challenging part in the project was to handle the collision which was solved by making rectangles around the ball and bars to detect the intersection point. At that time, the ball movement will be set to be the same with bars' speed. So it will create the vision that the ball is stopped by the bar.

REFERENCES

[1] Apress: Beginning Android Game,Mario Zechner. Chapter 3, Page 51

[2] Windows Phone Design System, Code Name Metro, Official PDF Document. October 8, 2012 http://go.microsoft.com/fwlink/?LinkID=189338

[3] Windows Phone 7 tombstoning explained, by Justin James. October 8, 2012 http://www.techrepublic.com/blog/smartphones/windows-phone-7-tombstoning-explained/2789

[4] Hardware Specification for Windows Phone, Official PDF Document. October 8, 2012. http://msdn.microsoft.com/en-us/library/ff637514(v=vs.92).aspx

[5] Introduction to the C# language. Microsoft official document. October 8, 2012 http://msdn.microsoft.com/library/z1zx9t92.aspx

[6] [7] Getting start with the .NET framework, Microsoft online document. October 8, 2012 . http://msdn.microsoft.com/en-us/library/hh425099.aspx

[8] Windows Phone Game Development,Adam Dawes.Page 23.What is XNA

[9] Microsoft Press: Programing Windows Phone 7,Charles Petzold.Page 23.An XNA program for the phone

[10] Windows Phone Software Development Toolkit, Official PDF Document. October 8, 2012. http://www.microsoft.com/enus/download/details.aspx?displaylang=en&id=27570

[11] Microsoft Express Book: Programming Windows Phone, Charles Petzold. Chapter 5: Sensor and Devices.Page 80.

[12] The Windows Phone Developer. October 8, 2012 http://windowsteamblog.com/windows_phone/b/wpdev/archive/2010/09/08/using-the-accelerometer-on-windows-phone-7.aspx

[13] Microsoft Express Book: Programming Windows Phone, Charles Petzold. Chapter 3, an introduction to Touch, Page 47

[14] Microsoft Express Book: Programming Windows Phone, Charles Petzold, Isolated storage, Page 126

[15] [16] Application certification requirement for Windows Phone, Microsoft official document. October 8, 2012 http://msdn.microsoft.com/en-us/library/hh184843(v=vs.92).aspx

APPENDICES

Appendix 1 MainGameScreen code

Appendix 2 Rapid Roller screenshots

Appendix 3 Link to Rapid Roller on Windows Phone market
http://www.windowsphone.com/en-GB/apps/6fe5fa07-ea6b-4ae4-b15b-d9f344235d07

APPEDIX 1

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Input.Touch;
using Microsoft.Xna.Framework.Media;
using Microsoft.Devices.Sensors;
using System.Windows;
using System.IO.IsolatedStorage;
using System.IO;

namespace Thesis
{
    public class MainGameScreen : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        Background _background;
        Ball _ball;
        Border _border;
        Pause _pause;
        int _currentScore = 0;
        int _highestScore = 0;
        Boolean _mute = false;
        Song _musicBackground;
        SoundEffect _hitSound;
        Gameover _gameover;
        Boolean _canPlayMusic = false;

        #region Notification player about the music
        private void CheckMusicCanPlay()
        {
            // if player does not listening to music
            //=> the game has conctrol of mediaplayer
            if (MediaPlayer.GameHasControl)
            {
                _canPlayMusic = true;
                // play music background
                LoopMusic(_musicBackground);
            }
            else
            {
                if (MessageBox.Show("Do u want to stop listening to
music?", "You are listening to Music", MessageBoxButton.OKCancel) ==
MessageBoxResult.OK)
                {
                    // if player click ok.
                    // stop current music and play game music
                    MediaPlayer.Stop();
```

```
                    _canPlayMusic = true;
                    // play music background
                    LoopMusic(_musicBackground);
                }
            }
        }
        #endregion

        #region Restart game
        private void Restart()
        {
            _ball.LoadContent(Content);
            _gameover.LoadContent(Content);
            _pause.LoadContent(Content);
            _currentScore = 0;
            _bars = new List<Bar>();
            OpenSaveFile();

        }
        #endregion

        #region High Score
        private void OpenSaveFile()
        {
            // Get the place to store data
            using (IsolatedStorageFile isf =
                IsolatedStorageFile.GetUserStoreForApplication())
            {
                if (!isf.FileExists("HighScore.dat"))
                {
                    // If the file does not exist, create it to save the
highscore data
                    using (IsolatedStorageFileStream isfs =
                        isf.CreateFile("HighScore.dat"))
                    {
                        using (StreamWriter writer = new
StreamWriter(isfs))
                        {
                            writer.WriteLine(_highestScore.ToString());
                            writer.Flush();
                            writer.Close();
                        }
                    }
                }
                else
                {
                    //if the file exist, open file
                    //and save so _highestScore
                    using (IsolatedStorageFileStream isfs =
                            isf.OpenFile("HighScore.dat",
FileMode.Open))
                    {
                        // Get the stream to read the data
                        using (StreamReader reader = new
StreamReader(isfs))
                        {
                            // Read the highscores
```

```
                            while (!reader.EndOfStream)
                            {
                                _highestScore =
int.Parse(reader.ReadLine());
                            }
                            reader.Close();
                        }
                    }
                }
            }
        }

        private void SaveHighScore()
        {
            // if the current score is bigger
            //then highest score, then save it
            if (_currentScore > _highestScore)
            {
                using (IsolatedStorageFile isf =
IsolatedStorageFile.GetUserStoreForApplication())
                {
                    using (IsolatedStorageFileStream isfs =
isf.OpenFile("HighScore.dat", FileMode.Open))
                    {
                        using (StreamWriter writer = new
StreamWriter(isfs))
                        {
                            writer.WriteLine(_currentScore.ToString());
                            writer.Flush();
                            writer.Close();
                        }
                    }
                }
            }
        }

        #endregion

        #region Looping music background
        private void LoopMusic(Song song)
        {
            if (_canPlayMusic)
                if (!_mute)
                {
                    MediaPlayer.Play(song);
                    MediaPlayer.IsRepeating = true;
                }
        }
        #endregion

        #region Play Sound Effect
        private void PlaySoundEffect(SoundEffect soundEffect)
        {
            if (_canPlayMusic)
                if (!_mute)
                    soundEffect.Play();
        }
```

```
        #endregion

        #region Bar Generator and Update
        // list to keep bars
        List<Bar> _bars = new List<Bar>();
        // set previousTimeSpan to zero
        TimeSpan _previousTimeSpan = TimeSpan.Zero;
        // set nextTimeSpan to 2 second
        TimeSpan _nextTimeSpan = TimeSpan.FromSeconds(2f);
        Random _random;
        private void BarGenerate()
        {
            Bar _bar1 = new Bar();
            Bar _bar2 = new Bar();
            _random = new Random();
            // first bar position is randomly
            Vector2 _barPostiion1 = new Vector2(_random.Next(-400,
GraphicsDevice.Viewport.Width - 400), 810);
            // second bar position is at the same line, but separate
with first bar by 430 pixel
            // This create a constantly space between 2 bars which the
ball can go through
            Vector2 _barPostiion2 = new Vector2(_barPostiion1.X + 430,
810);
            _bar1.LoadContent(Content, _barPostiion1);
            _bar2.LoadContent(Content, _barPostiion2);
            // add 2 bar to bars list
            _bars.Add(_bar1);
            _bars.Add(_bar2);
        }
        private void BarUpdate(GameTime _gametime)
        {

            // regenerte bar every 2s
            if (_gametime.TotalGameTime - _previousTimeSpan >=
_nextTimeSpan)
            {
                _previousTimeSpan = _gametime.TotalGameTime;
                BarGenerate();
            }

            // when bar goes out of the screen , remove it
            for (int i = 0; i < _bars.Count(); i++)
            {
                if (_bars[i].Position.Y < -50)
                    _bars.RemoveAt(i);
                _bars[i].Update();
            }
        }
        #endregion

        #region Accelerometer Fucntion

        Accelerometer accelerometer = new Accelerometer();
        double X;
        //double Y;
        //double Z;
```

```
        void AccelerometerReadingChanged(object sender,
AccelerometerReadingEventArgs e)
        {
            Deployment.Current.Dispatcher.BeginInvoke(() =>
NewReading(e));
        }
        void NewReading(AccelerometerReadingEventArgs e)
        {
            X = e.X;
            //Y = e.Y;
            //Z = e.Z;
        }

        #endregion

        #region Collision detect

        private void CollisionDetect()
        {
            Rectangle rectangle1;
            Rectangle rectangle2;
            // rectangle1 cover the ball
            rectangle1 = new Rectangle((int)_ball.Position.X,
(int)_ball.Position.Y, _ball.BallTexture.Width,
_ball.BallTexture.Height);

            for (int i = 0; i < _bars.Count; i++)
            {
                // rectangle2 cover bars
                rectangle2 = new Rectangle((int)_bars[i].Position.X +
20, (int)_bars[i].Position.Y, _bars[i].BarTexture.Width - 8,
_bars[i].BarTexture.Height);
                if (_bars[i].Active)
                {
                    if (!_bars[i].Hit)
                    {
                        // if the ball position is
                        // smaller bar position
                        // do not check becasue the
                        // ball already pass through the bar
                        if (_ball.Position.Y <= _bars[i].Position.Y)
                            // if rectangle1 intersect rectangle2
                            // => ball hit bar
                            if (rectangle1.Intersects(rectangle2))
                            {
                                // play hit sound
                                PlaySoundEffect(_hitSound);
                                _bars[i].Hit = true;
                                // ball speed take the bar speed value.
                                // so the ball moving up
                                _ball.Speed = _bars[i].Speed;
                                // keep the ball texture
                                // not to step on bar texture
                                _ball.Position.Y =
MathHelper.Clamp(_ball.Position.Y, _bars[i].Position.Y - 18,
_bars[i].Position.Y - 16);
                            }
```

```
                    }
                    else
                    {
                        if (!rectangle1.Intersects(rectangle2))
                        {
                            // do not check this bar anymore
                            _bars[i].Active = false;
                            // turn ball speed to gavity
                            //=> ball continue drown down
                            _ball.Speed = _ball.Gavity;
                        }
                    }
                }
            }
        }

        #endregion

        #region Handle input touch

        private void HandleInput()
        {

            TouchCollection _touches = TouchPanel.GetState();
            if (_touches.Count > 0 && _touches[0].State ==
TouchLocationState.Pressed)
            {
                Microsoft.Xna.Framework.Point _touchPoint = new
Microsoft.Xna.Framework.Point((int)_touches[0].Position.X,
(int)_touches[0].Position.Y);

                if (!_gameover.active)
                {
                    #region Pause Panel Input
                    // if player touch pause button
                    if (_pause.RecPauseButton().Contains(_touchPoint))
                    {
                        if (!_pause.active)
                            _pause.active = true;
                        else
                            _pause.active = false;
                    }

                    if (_pause.active)
                    {
                        // if player touch menu button
                        if (_pause.RecMenu.Contains(_touchPoint))
                        {
                            accelerometer.Stop();
                            this.Exit();
                        }
                        // if player touch sound button
                        if (_pause.RecSound.Contains(_touchPoint))
                        {
                            // pause and resume mmusic and sound
                            if (!_mute)
                            {
```

```
                        _mute = true;
                        MediaPlayer.Pause();

                    }
                    else
                    {
                        _mute = false;
                        MediaPlayer.Resume();
                    }
                }
                // if player touch resume button
                if (_pause.RecResume.Contains(_touchPoint))
                {
                    _pause.active = false;
                }

            }

            #endregion
        }
        else
        {
            #region Gameover Panel Input
            // if player touch play again button
            if (_gameover.RecAgain.Contains(_touchPoint))
            {
                SaveHighScore();
                Restart();
            }
            // if player touch menu button
            if (_gameover.RecMenu.Contains(_touchPoint))
            {
                SaveHighScore();
                accelerometer.Stop();
                this.Exit();
            }
            // if player touch quit button
            if (_gameover.RecQuit.Contains(_touchPoint))
            {
                SaveHighScore();
                accelerometer.Stop();
                this.Exit();
            }
            #endregion
        }
    }
}

#endregion

public MainGameScreen()
{

    graphics = new GraphicsDeviceManager(this);

    // full screen
    graphics.IsFullScreen = true;
```

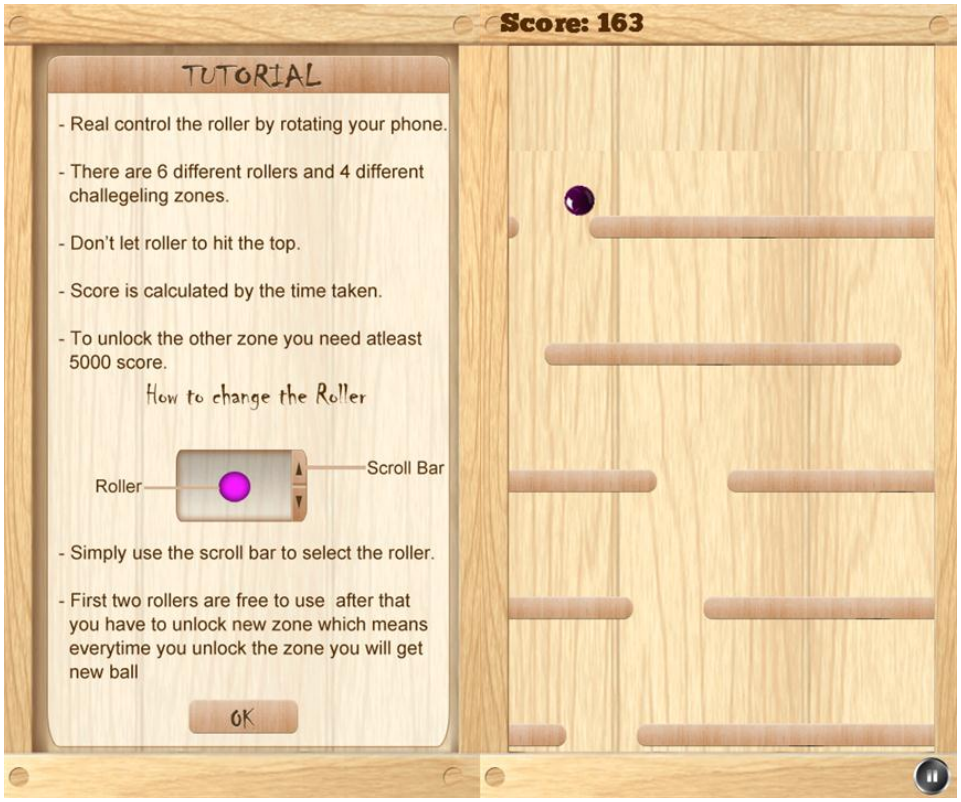
```
            graphics.PreferredBackBufferHeight = 800;
            graphics.PreferredBackBufferWidth = 480;


            Content.RootDirectory = "Content";

            // Frame rate is 30 fps by default for Windows Phone.
            TargetElapsedTime = TimeSpan.FromTicks(333333);

            // Extend battery life under lock.
            InactiveSleepTime = TimeSpan.FromSeconds(1);
        }

        protected override void Initialize()
        {
            _background = new Background();
            _ball = new Ball();
            _border = new Border();
            _pause = new Pause();
            _gameover = new Gameover();

            // start the accelerometer
            accelerometer.ReadingChanged += new
EventHandler<AccelerometerReadingEventArgs>(AccelerometerReadingChanged)
;
            accelerometer.Start();

            OpenSaveFile();
            base.Initialize();
        }

        protected override void LoadContent()
        {
            //Create a new SpriteBatch, which can be used to draw
textures.
            _background.LoadContent(Content);
            _ball.LoadContent(Content);
            _border.LoadContent(Content);
            _pause.LoadContent(Content);
            _musicBackground = Content.Load<Song>("MusicBackground");
            CheckMusicCanPlay();
            _hitSound = Content.Load<SoundEffect>("BallHit");
            _gameover.LoadContent(Content);
            spriteBatch = new SpriteBatch(GraphicsDevice);
        }

        protected override void Update(GameTime gameTime)
        {
            // back button
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
ButtonState.Pressed)
            {
                if (_pause.active)
                {
                    if (!_pause.active)
                        _pause.active = true;
```

```
            else
                _pause.active = false;
        }
        if (_gameover.active)
        {
            SaveHighScore();
            Restart();
        }
    }

    // stop game loop if gameover active
    if (!_gameover.active)
    {
        // pause the game loop if pause active
        if (!_pause.active)
        {

            #region Acceleremeter update ball position
            if (X < 0)
            {
                // rotate the ball picture to left
                _ball.Rotation -= 0.3f;
                // move the ball position to left
                _ball.Position.X += (float)X * 12;

            }
            else
            {
                // rotate the ball picture to right
                _ball.Rotation += 0.3f;
                // move the ball position to right
                _ball.Position.X += (float)X * 12;
            }
            #endregion

            // if the ball hit top. gameover active
            if (_ball.Position.Y <= 48)
                _gameover.active = true;

            _currentScore++;   // score increase by time
            _background.Update();
            _ball.Update();
            CollisionDetect();
            BarUpdate(gameTime);

        }
        _pause.Update();
    }
    _gameover.Update();
    HandleInput();
    base.Update(gameTime);
}

protected override void Draw(GameTime gameTime)
{
    GraphicsDevice.Clear(Color.CornflowerBlue);
    spriteBatch.Begin();
```

```
            // draw background
            _background.Draw(spriteBatch);

            // draw bars
            foreach (Bar b in _bars)
                b.Draw(spriteBatch);

            //draw ball
            _ball.Draw(spriteBatch);
            _border.Draw(spriteBatch);
            _pause.Draw(spriteBatch);
            _gameover.Draw(spriteBatch);

            // display current score and highest score when game over
            if (_gameover.active)
            {

spriteBatch.DrawString(Content.Load<SpriteFont>("GameOverFont"),
_currentScore.ToString(), new Vector2(270, _gameover.position.Y - 56),
Color.Black);

spriteBatch.DrawString(Content.Load<SpriteFont>("GameOverFont"),
_highestScore.ToString(), new Vector2(270, _gameover.position.Y - 56 +
47), Color.Black);
            }

spriteBatch.DrawString(Content.Load<SpriteFont>("ScoreFont"), "Score: "
+ _currentScore.ToString(), new Vector2(20, 2), Color.Black);
            spriteBatch.End();
            base.Draw(gameTime);
        }

    }
}
```

APPEDIX 2


RAPID ROLLER
Start
H.Score
Options
Sound
Quit
Tutorial
SELECT LEVEL
ZONE 1
ZONE 2
ZONE 3
ZONE 4
OK
SELECT ROLLER
OK


Score: 163
TUTORIAL
- Real control the roller by rotating your phone.
- There are 6 different rollers and 4 different challegeling zones.
- Don't let roller to hit the top.
- Score is calculated by the time taken.
- To unlock the other zone you need atleast 5000 score.
How to change the Roller
Roller
Scroll Bar
- Simply use the scroll bar to select the roller.
- First two rollers are free to use after that you have to unlock new zone which means everytime you unlock the zone you will get new ball
OK