1. **Specification**

The game is called “Mario Fun Game”. In this game Mario is the main figure. Mario makes is movement left to right, all these movements are done by Brick in other to hit various fruits that are already at the ground also when Mario hit the fruit it can also reappear. This is beneficial to Mario as hitting the fruits will add to Mario scores. While the timer is counting down, Mario must move away from the enemy which is call the ‘brick’ to prevent him from losing and it can eventually cause the game to be over.

**1.2Features**

1. Mario
2. Bricks
3. Fruits

**1.3Functional Requirements**

1. Move-Left by Mario
2. Move-Right by Mario
3. Fruits
4. Bricks
5. Lives
6. Score
7. Name
8. Instruction
9. body collision detection

Non-functional requirements.

1. Sound
2. Level

**2.1 Analysis**

|  |  |
| --- | --- |
| Inputs: | Key’s to control Mario  Arrow right – RIGHT.  Arrow left – LEFT. |
| Outputs: | Fruits  bricks (enemy)  Score  Game Over if the tom should hit the bricks three time.  Game instructions on start page |
| Processes: | A function to display tom within map grid.  Wait for key press to control Mario movement.  A function to randomly generate fruit  A function to randomly generate brick (enemy)  For statement to increase score b 1 if Tom should hit the fruit.  A counter is set to count the time down for 60 seconds  A function to display game and final score when, or maybe the player want to quit the game. |

**2.2 Analysis**

Use case.

|  |  |
| --- | --- |
| Score | Score is added or increases when Mario hits fruits. |
| Actor | The name of the actor (Mario) who participates in the game |
| Brief description | Score increases when Mario hits fruits moving from left to right. |
| Precondition | The score at default is 0. |
| Basic Flow | 1. The player starts the game 2. The Score is at 0 3. Player moves left, right and can jump via a key press 4. Mario hits fruits as he makes moves to gain score. 5. Score increases by 1 on each successful hit. |
| Post Condition | Score increases as Mario hits fruits. |

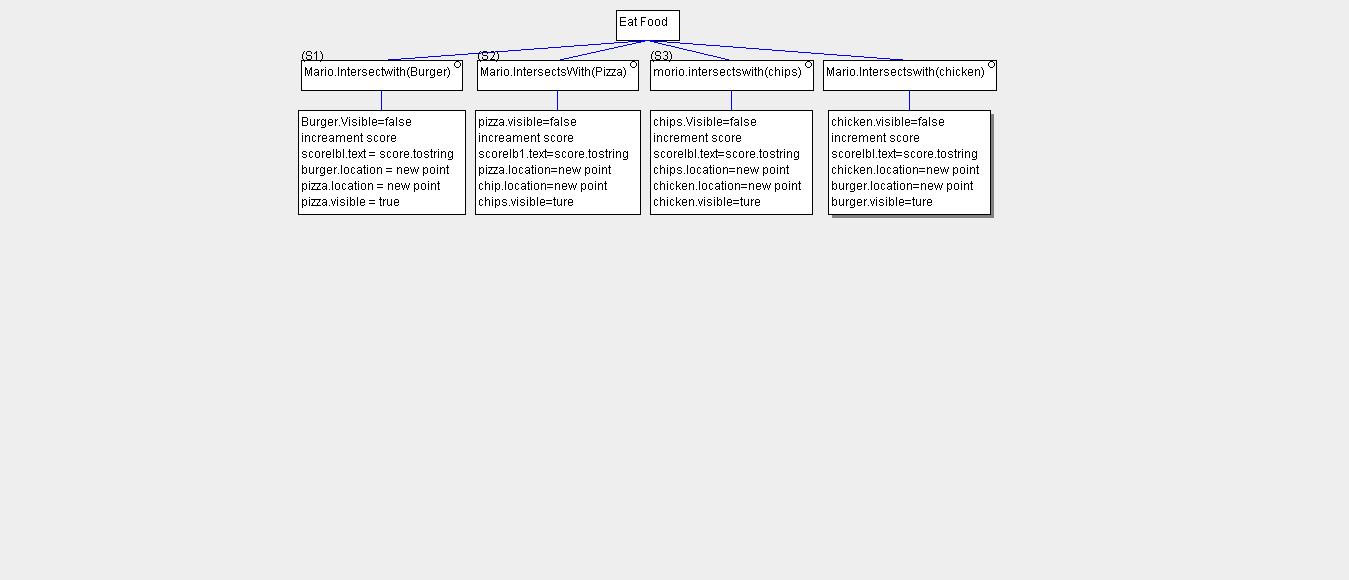
**3. Design**

**3.1 Top Level Overview**

The system has the following top-level stages of execution:

1. Wait for game to run by pressing the start menu or (debugging).
2. Click on start menu to play.
3. Read the instruction to understand the game procedure.
4. Press the key to move the actor (Mario).
5. Display of food and bricks
6. Score increase by 1 when tom hit the food.
7. Live set to 3
8. Display game over
   1. **Flowchart**

**A JSP diagram to represent food and increment in score when Mario hit the food**.



**3.3“Pseudo-code for Mario Game”**

The complete pseudo code for Tom fun game.

|  |  |  |
| --- | --- | --- |
| **Function:** | **Display Menu** | |
|  | Display “Game start” | |
|  | Display “Instruction” | |
|  |  | |
|  | Display “Quit” | |
|  | DO | |
|  | Variable choice = user input | |
|  | IF input = press enter THEN | |
|  | Display “Mario Game” | |
|  | Call function move Mario | |
|  | Move Tom Right  If key= Arrow right | |
|  | Call function move left  If Key = Arrow left | |
|  | While control key=True | |
|  | **Function: Get score**  call function is set score=20  Tom should hit fruit  Score increased by 1  If Score remain at default  **Function: Get lives**  Call function set live =3  Tom hit the brick  Than score reduce by 1  Else  Press key = Enter  Call function to move actor again  Tom hit the brick  Score reduce by 1  Else  Press key = Enter  Call function to move the again  Tom hit the brick  Call function to show Game Over  Mario is dead!!!  Else  Call function key = Enter  Take player back to the Menu  If  Start playing. |
|  |  |
|  |  |

* 1. Interface design

The game interface consist of

* The actor (Mario)

The actor is control by the prayer which move left and right.

* Food /brick

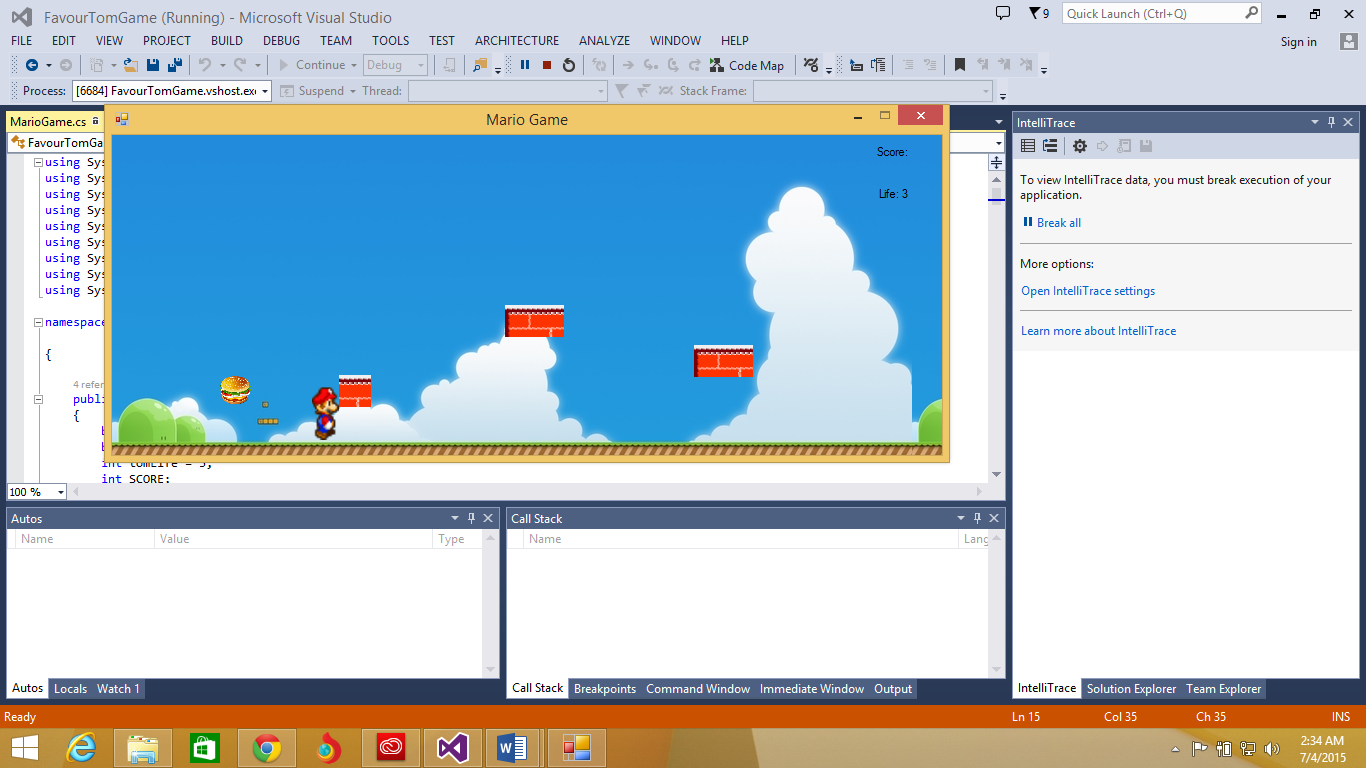
The food is one of the function requirement of the game because without the food the game can function well. Also the brick is the enemy and when tom hide the rock its end the game automatically.

* Live

This is to give the player a clear understand on how many live the game have.

* Score

The score show the number of times Mario hit the food.



**4.1 Implementation**

This game was implemented in C# using Visual Studio 2013

**4.2 Classes**

Mario game -Used as the main application class. Derived from Windows. Forms

4.3 Objects

|  |  |
| --- | --- |
| Food | Randomly generated within the map grid and when mairo hit the food score increment |
| Brick | The brick represent the enemy and it is randomly generated in map gird by falling randomly. |
| Actor (Mario) | Mario Is the main figure of the game control |

4.4 Example of code

The following code is run whenever the user pressed the keys to control Mario.

private void Form2\_KeyDown(object sender, KeyEventArgs e)

{

Point MoveMario;//decare a point for player picturebox

MoveMario = MarioCharacter.Location;//intilize the point with the picturebox location

if (e.KeyCode == Keys.Left)//using the leftarrow key to move the player

{

MoveMario.X -= 10;//decrement the X coordinate

MarioCharacter.Location = MoveMario;//set the new location to the new coordinate

}

if (e.KeyCode == Keys.Right)//using the leftarrow key to move the player

{

MoveMario.X += 10;//Increment the X coordinate

MarioCharacter.Location = MoveMario;//set the new location to the new coordinate

}

if (MoveMario.X <= -10) //decrement the X coordinate

{

MoveMario.X = 810;// position for mario in aixs

MarioCharacter.Location = MoveMario;//set the new location to the new coordinate

}

if(MoveMario.X >= 811)// position for mario in aixs

{

MoveMario.X = -11;//decrement the X coordinate

MarioCharacter.Location = MoveMario;//set the new location to the new coordinate

}

The following code is run to generate food within the map grid.

private void eatFood()

{

if (MarioCharacter.Bounds.IntersectsWith(Burger.Bounds)) // if mario bounds intersect with the food bounds than :

{

Burger.Visible = false;

SCORE++; //Increment the score.

scoreLbl.Text = "Score: " + SCORE.ToString();

Burger.Location = new Point(598, 135);//Change the position of the Burger in a manner that the Tom Sprite will not touch it even though it is invisible.

Pizza.Location = new Point(598, 240);//Change the position of the pizza in a manner that the Tom Sprite will not touch it even though it is invisible.

Pizza.Visible = true;

}

if (MarioCharacter.Bounds.IntersectsWith(Pizza.Bounds))

{

Pizza.Visible = false;

SCORE++; //Increment the score.

scoreLbl.Text = "Score: " + SCORE.ToString();

Pizza.Location = new Point(598, 135);//Change the position of the pizza in a manner that the Tom Sprite will not touch it even though it is invisible.

Chips.Location = new Point(343, 240);//Change the position of the

Chips.Visible = true;

}

if (MarioCharacter.Bounds.IntersectsWith(Chips.Bounds))

{

Chips.Visible = false;

SCORE++; //Increment the score.

scoreLbl.Text = "Score: " + SCORE.ToString();

Chips.Location = new Point(598, 135);//Change the position of chips in a manner that the Tom Sprite will not touch it even though it is invisible.

Chicken.Location = new Point(12, 240);//Change the position of the.

Chicken.Visible = true;

}

if (MarioCharacter.Bounds.IntersectsWith(Chicken.Bounds))

{

Chicken.Visible = false;

SCORE++; //Increment the score.

scoreLbl.Text = "Score: " + SCORE.ToString();

Chicken.Location = new Point(598, 135);//Change the position of the.

Burger.Location = new Point(615, 240);//Change the position of the

Burger.Visible = true;

}

}

The following code is run to generate brick within the map grid and also their location.

private void moveFallingObjects()

{

//Making the bricks to fall down as the time ticks.

brickPic1.Location = new Point(brickPic1.Location.X, brickPic1.Location.Y + 10);

brickPic2.Location = new Point(brickPic2.Location.X, brickPic2.Location.Y + 10);

brickPic3.Location = new Point(brickPic3.Location.X, brickPic3.Location.Y + 10);

if (brickPic1.Location.Y == 330)

{

brickPic1.Location = new Point(brickPic1.Location.X - 30, 0);

if (brickPic1.Location.X <= 0)

brickPic1.Location = new Point(185, 0);//lcation

}

}

if (brickPic2.Location.Y == 330)//location for Y aixs brickpic2

{

brickPic2.Location = new Point (brickPic2.Location.X - 3, 0);

if (brickPic2.Location.X <= 169)

{

brickPic2.Location = new Point (744, 0);//create new position location for brickpic3.

}

}

if (brickPic3.Location.Y == 330)// location for Y aixs for brickpic3

{

brickPic3.Location = new Point (brickPic3.Location.X + 50);//create new position location for brickpic3.

if (brickPic3.Location.X >= 776)//location for brickpic3 X aixs

{

brickPic3.Location = new Point (347, 0);//create new position location for brickpic3.

}

}

}

5. Testing

* 1. Black Box Testing

The application was give to 4 user who had no knowledge of how the application works

The following was reported

* Would have been better to include a levels
* Include a bourse food that makes the score to go higher.
* Include sound to the game so the game may be more enjoyable.
* There should be increases in the number of bricks and food
* Include a boundary within the X coordinate, so the Mario cannot move around the map grid

**5.2 Function Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Scenario | Test Data | Expected Result | Actual Result | Notes |
| Movement | Press(arrow right)-right  Press (arrow left)-left. | Mario moves when player’s presses the key | As expected,  Movement work with caps lock on | When the player stop pressing the key Mario stop at that moment. |
| Food generation |  | Generate food within map grid | As expected |  |
| Brick collision | Move Mario to hit the brick | Print out you have been hit unit is up to 3 time print game over | As expected | when Mario hit the brick there is a pause in the game |
| Food increment | Move Mario to hit the food | Increase score, and randomly generate another food | Regenerates food, Increases in score. |  |

**5.3 critique**

|  |  |
| --- | --- |
| Things i would have improved if I had more time | * Inclusion of different levels and difficulties that would include more sprites such as poisonous food which snake should avoid. * Player would be able to restart game after game is over. * Included a pause function to enable player to pause the game. * Worked on a better algorithm to increase snake length, bending during movement and body collision. |
| Things included after testing | * Inclusion of sound file when snake hits the map borders |

**7. Evaluation**

The program meets many of the requirements in the specification, however it does not:

* Body collision detection
* Score
* Live

**8. References**

-How to move a character <https://msdn.microsoft.com/en-us/library/system.windows.forms.control.move(v=vs.110).aspx>

-Read test file string array C# <https://msdn.microsoft.com/en-us/library/s2tte0y1(v=vs.110).aspx>