“Tiler”

“2020 The End of the World”

A simple Falling Block survival game to pass the time

Alex Brady

20438266

CS171 Computer Systems I

B.Sc.

Computer Science and Software Engineering

Maynooth University
National University of Ireland Maynooth

Gallon

Carrickaboy

Cavan, Co. Cavan

Ireland

Declaration

I hereby certify that this material, which I now submit for assessment as part of CS171 Computer Systems module, is entirely my own work and has not been taken from the work of others - save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: **Alex Brady**

Acknowledgements

I would like to thank my professors Peter Mooney, John McDonald, Joe Timoney and Stephen Brown for helping to inspire me to create this small game and to some of the other members of my class for helping me brainstorm several ideas for this project, before I ultimately decided on the final idea of a falling block game.

Abstract

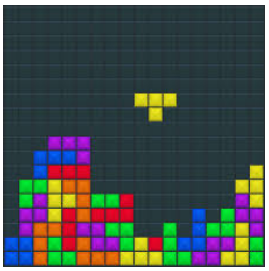
This report describes the development process of a program designed to be a small block falling survival game. The program created works by having small rectangular objects fall from the top of the screen to the bottom, whilst the user controls a slender rectangle at the bottom of the screen, the slender rectangle is meant to symbolise a person and the falling blocks are meant to be meteors. The aim of the game is to control the person by clicking and dragging it out of the way of the falling meteors if hit the game ends. The score is recorded by how many blocks u dodge and is displayed at the top left of the screen and then again in the centre of the screen when the game is over.2

1 Introduction

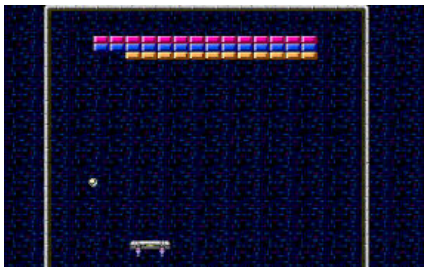
The aim of this project was to create a small java based game in Processing which had objects fall from the top of the screen to the bottom whilst the user controlled a small rectangle at the bottom and tries to avoid the falling objects. The falling objects within the game are known as meteors and the player controlled object is known as a person. With each meteor dogged the score is increased by one but if hit by a meteor the game ends. The higher your score the progressively more difficult the game gets. The game is meant to be just a simple thing used to pass the time when bored.

2 Research

There are many different games that use similar mechanics to this game such as in Tetris when the blocks are falling down the screen before settling into place. Or even more complex games such as the mobile game Subway Surfers which in simple terms has the same concept of avoid the objects and the longer you live the higher your score gets but the higher the difficulty also gets. There is a significant amount of information in academic literature that talk about the collision of items in java games and how different outcomes can come from the collision of objects depending on the planned outcome you wish to see from your code such as sticking to the spot it lands on or bouncing in a different direction which can be seen in things such as block breaking games or pong.



Tetris falling mechanics

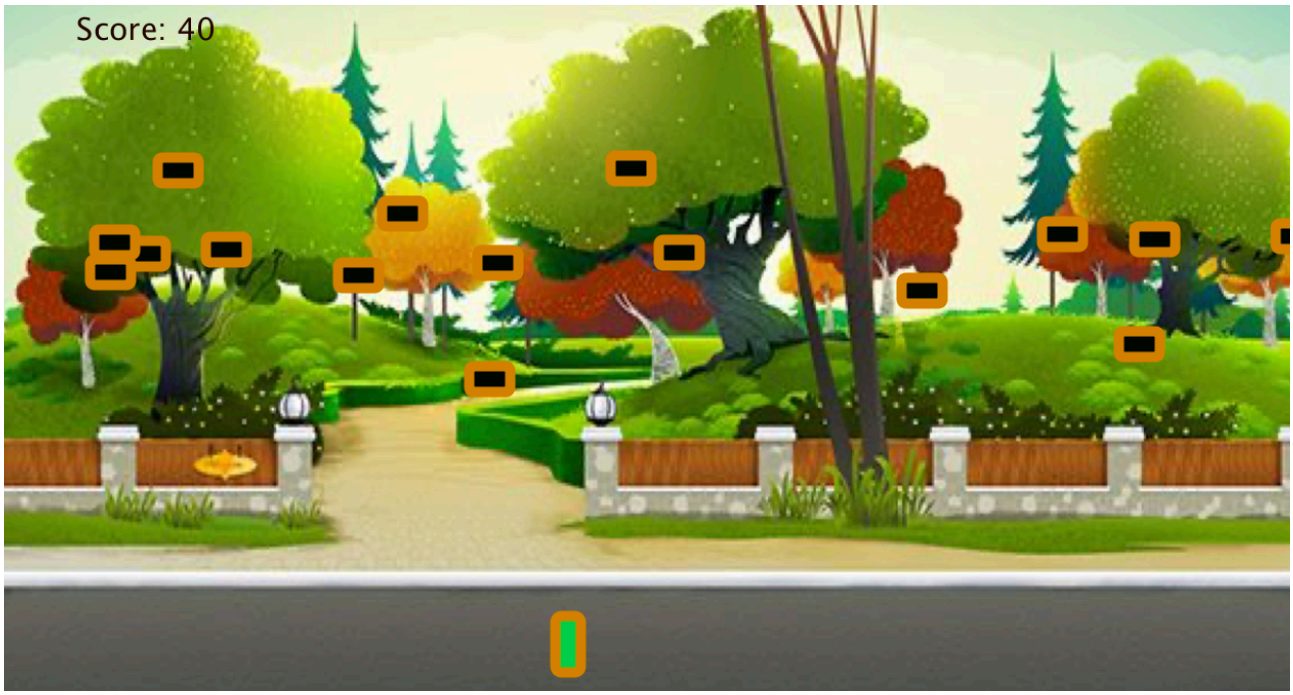


Block Breaker collision mechanics

3 Technical background and problem statement

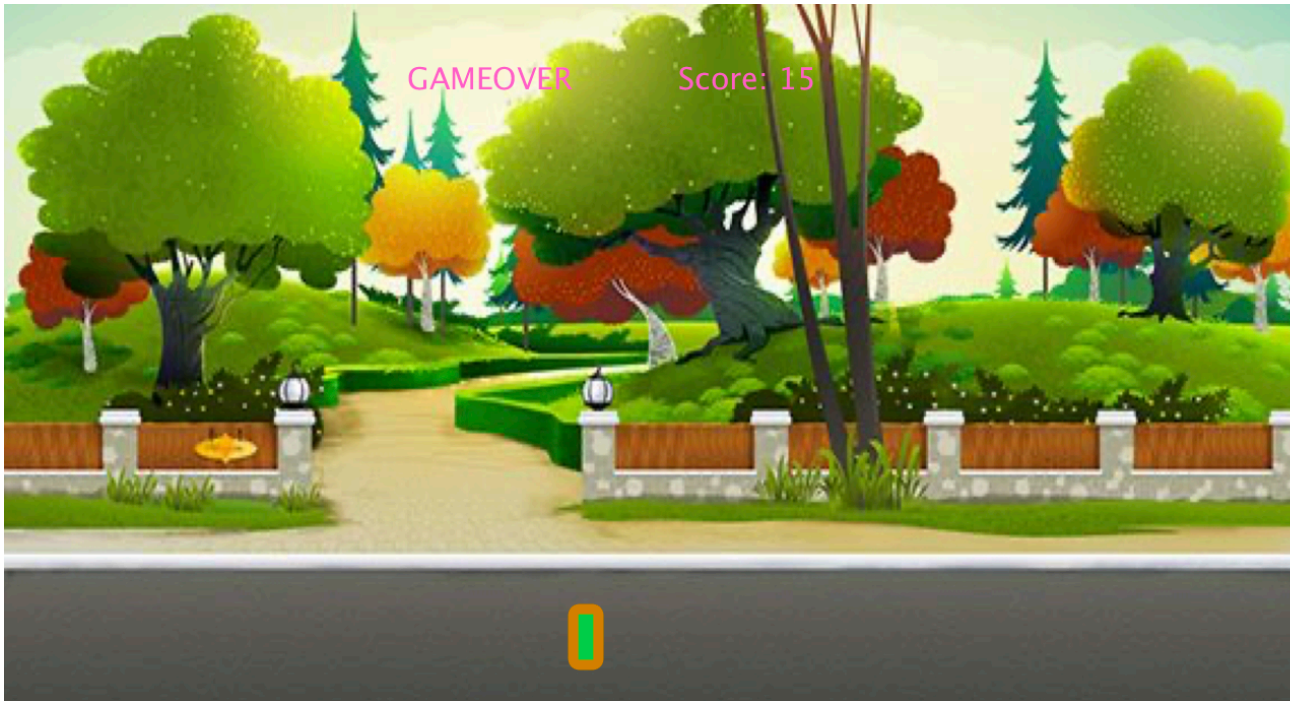
The simplest form of what is being shown within the game would be to have the meteors fall in the same places over and over again using hard coded locations. But to achieve the end goal I wanted to have the blocks fall in random locations with increasing levels of difficult as time progresses having more and more meteors fall leaving less room for the person to move too to be safe.

In this scenario the size of the meteors are all the same and there speed is the same but the height in which they can spawn is slightly varied. The speed they fall is slightly effected in a gravity type way too.



4 Solution

The program developed contains two data structures one for the meteors and one for the person. The background image is stored with the game and is loaded as the code starts to run. The meteors make use of fundamentals of a method that was found in a YouTube video, in the way they fall and gradually increase speed giving the game a gravity like effect on the meteors. The collision detection between the person and the meteors is stirred in a boolean loop and checks to see if any part of the person and meteor have touched if so the game ends and proceeds to show a game over screen and the score reached.



5 Evaluation

The program was tested several times moving the person in all different directions and making contact with the meteors in all different ways possible making sure the hit detection was working as intended, it was also tested by staying alive as I could and seeing if the game still worked as intended with an extreme amount of meteors on the screen.

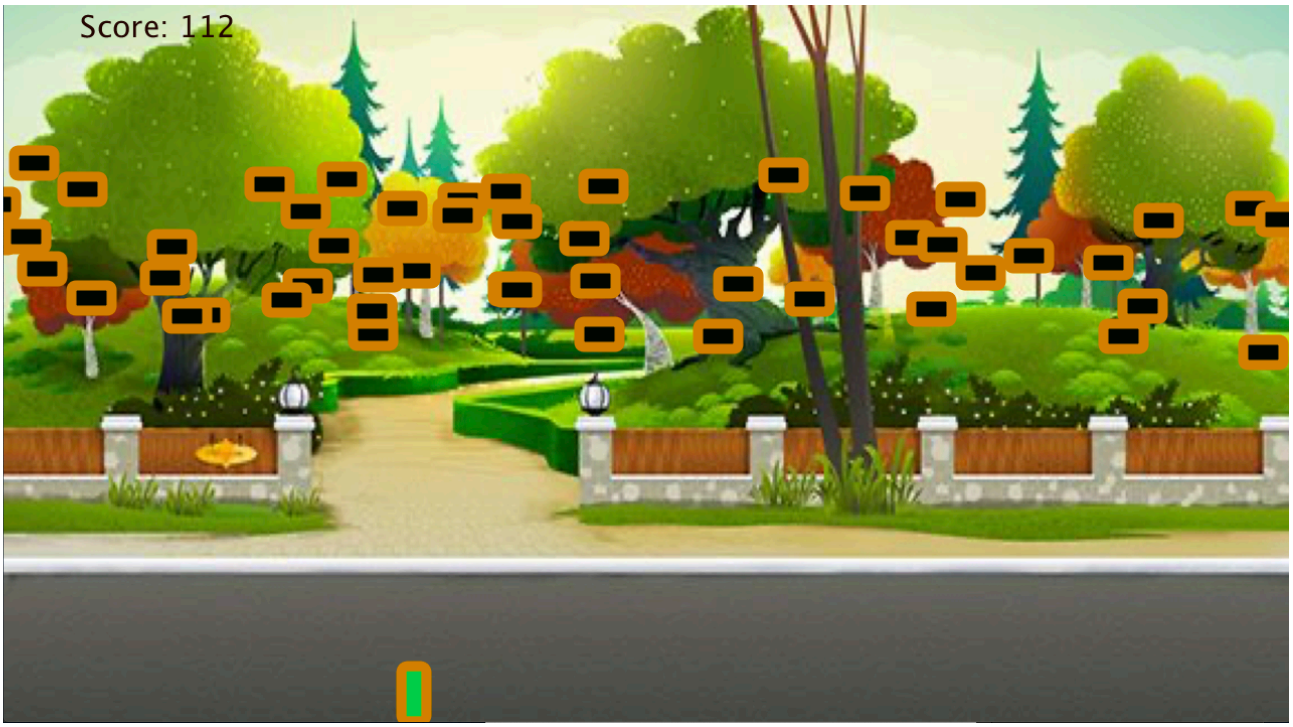


Figure 5.1 Several test were carried out to make sure the score was also counting correctly and wasn't missing or adding any points. To test this I recorded the game screen with my phone and caused the game to end after the first wave of meteors, I then proceed to count how many fell and compare it to the score I got and the two were the same.

6 Conclusion

A Program has been created that is fun and effective at passing time in the form of gaming. A future development required is the implementation of an actual character image for the players controlled user and the same for the meteors. The code could also be extended to have a lives system giving the player a better chance at surviving for long.

Finally if the program were developed far enough a level/world system could be implemented where if the player reaches a required score in a level they would progress to the next area.

References

[1] Website- https://forum.processing.org/two/discussion/13721/#Comment_56423

[2] Video- https://youtu.be/KkyIDI6rQJI

[3] Book- Programming Fundamentals Using **Java**: A **Game** Application Approach

by McAllister, William; Fritz, S. Jane

[4] Website- https://stackoverflow.com/questions/63630412/problems-on-making-the-game-falling-blocks-on-processing

Appendix A Source code listing

```
PImage img;


//player
int playerXCor = 600;
int playerYCor = 590;
int playerWidth = 25;
int playerHeight = 55;
//meteor
int difficulty = 10;
int limit = 8;
double score = 0;
Meteor[] meteor;
boolean isCollided = false;



void initMeteor(int xMin, int xMax, int yMin, int yMax, int num){
  meteor = new Meteor[num];
 //positioning of the meteors
  for(int i = 0; i < meteor.length; i++){
     int x = (int)random(xMin, xMax);
     int y = (int)random(yMin, yMax);
```

```
    meteor[i] = new Meteor(x, y, 40, 25);
  }
}

void setup(){
  //fullScreen();
  size(1280,720);
  //rameRate(60);
  img = loadImage("bg.png");
  initMeteor(-100, width + 20, -250, -80, difficulty);



}

void draw(){

  image(img,0,0); // Using instead of background();
  drawPlayer();
  //checks to see if collision happened if no the game continues
  if(!isCollided){
    moveMeteor();
    if(score > limit && score < limit + 8){
      initMeteor(-100, width + 20, -260, -80, difficulty); difficulty += 8; limit += 20;
    }
  }
  //else yes the game ends
 else{
   lives=0;
   fill(255,110,199);
   text("GAMEOVER          Score: "+(int)score, 400, 100); //end of game text
 }



}

void moveMeteor(){
    for(int i = 0; i < meteor.length; i++){
```

```
      if(meteor[i].yCor > height){
        meteor[i].yCor = -20;
      }
      meteor[i].display();              //taken from meteors tab
      meteor[i].drop(random(5, 15));     //taken from meteors tab
  //collision checking
  // should be in its own loop but causes game to slow down
      boolean conditionXLeft = meteor[i].xCor + meteor[i].w >= playerXCor;
      boolean conditionXRight = meteor[i].xCor + meteor[i].w <= playerXCor + playerWidth + 4;
      boolean conditionUp =  meteor[i].yCor >= playerYCor;
      boolean conditionDown = meteor[i].yCor + meteor[i].h <= playerYCor + playerHeight;

      if(conditionXLeft && conditionXRight && conditionUp && conditionDown){
          isCollided = true;

      }

    }

  //score counter
   score += 0.1;

   fill(30, 5, 0);
   text("      Score: "+(int)score, 10, 40);
   textSize(30);
}

void drawPlayer(){
  stroke(204, 132, 0);
  strokeWeight(10);
  //shape and color of player controlled block
  fill(90, 200, 90);
  rect(playerXCor, playerYCor, playerWidth, playerHeight, 7);
}
```

```
//movement controls
void mouseDragged(){
  if(mouseX >= 0 && mouseX <= width - playerWidth + 1){
    playerXCor = mouseX;
  }
  if(mouseY >= 600 && mouseY <= height - playerHeight){
    playerYCor = mouseY;
  }

}

//meteor specifications and class
public class Meteor{

   int xCor;
   int yCor;
   int w;
   int h;

  Meteor( int xVal,int yVal, int wVal, int hVal){
    xCor = xVal;
    yCor = yVal;
    w = wVal;
    h = hVal;

  }
  //falling parameters
  public void drop(double speed){
    yCor += speed;
  }
  //disyplay function
  public void display(){
    fill(0, 9, 0);
    rect(xCor, yCor,w,h,5);
  }
}
```