spaceBox Game

Seyyare Ece Akdağ[1]

ABSTRACT

[1] Istanbul Technical University, Graduate School of Science, Engineering, and Technology, Department of Informatics, Architectural Design Computing, Istanbul, Turkey
[2] Istanbul Technical University, Faculty of Architecture, Department of Architecture, Istanbul, Turkey

The aim of this investigation is to explore the computational programming languages on the task to create a game in Processing language. A game algorithm is aimed to be designed so that the definitions like player, obstacles, place, and time are converted into computational representation in the scripting scene to cooperate to become a game. The inspiration of the game design for this project is the well-known mobile app game called “Flappy Bird”. The strategy for the game development has started with the idea of creating such a simple yet enjoyable arcade game. The game represents a possible coding simplicity for a beginner game developer, plus its testing procedure is not complex as multi-leveled more complex games, so that easy mistake detection is possible. The parallel design to the famous arcade game is developed in the way that one player would be moving forward on a two-dimensional scene set-up, and obstacles are going to be met by the player, and using the control keys on the keyboard, the player would escape the obstacles to collect more points. On such a development study, strategical features to be considered are emerged such as considering movement, speed, location on screen, gravity and so on. Concept-wise the game atmosphere is decided to be space and objects are remined as geometries, so the name of the game has become “spaceBox”. The main aim behind the conceptual idea was to make is simple to be parallel with the minimalism of the game algorithm, and create a neutral vibe for its target audience by not having negative connotations on any group of people from different counties, beliefs, genders etc.

Corresponding Author:
akdags22@itu.edu.tr

Akdağ, A. (2023). spaceBox Game. JCoDe: Journal of Computational Design

Keywords: Max. 5 keywords should be written in Calibri Light with 9.5-point size. Each keyword should be separated by a comma and must start with a capital letter.

spaceBox Game

Seyyare Ece Akdağ[1]

ÖZET

[1] İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Bilişim Anabilim Dalı, Mimari Tasarımda Bilişim, İstanbul, Türkiye
[2] İstanbul Teknik Üniversitesi, Mimarlık Fakültesi, Mimarlık Bölümü, İstanbul, Türkiye

Bu araştırmanın amacı, işleme dilinde bir oyun yaratma görevi üzerinden hesaplamalı programlama dillerini keşfetmektir. Oyuncu, engel, yer, zaman gibi tanımların komut dosyası oluşturma sahnesinde sayısal temsile dönüştürülerek oyun haline getirilmesi için bir oyun algoritması tasarlanmak istenmiştir. Bu proje için oyun tasarımının ilham kaynağı, "Flappy Bird" adlı tanınmış mobil uygulama oyunudur. Oyun geliştirme stratejisi, bu kadar basit ama eğlenceli bir atari oyunu yaratma fikriyle başlamıştır. Oyun, başlangıç seviyede işlem ve tasarım yapabilen bir oyun geliştiricisi için olası bir kodlama basitliğini temsil eder, ayrıca test prosedürü, çok seviyeli daha karmaşık oyunlar kadar karmaşık olmadığından, test etme prosedürü hızlı ve etkili bir şekilde hata saptama yapmaya yardımcı olmaktadır. İlham alınan bilindik atari oyununa paralel bir tasarım olarak, bir oyuncunun iki boyutlu bir sahne kurgusunda ilerlemesi ve engellerin oyuncu tarafından aşılması ve klavyedeki kontrol tuşlarını kullanması şeklinde bir öngörü ve algoritma şeması üzerinden oyunun temeli geliştirildi. Oyuncu puan toplama amacı doğrultusunda karşısına çıkan engellerden kaçarak oyunda hayatta kalacak şekilde bir altyapı kurgulanmıştır. Böyle bir geliştirme çalışmasında hareket, hız, ekrandaki konum, yerçekimi gibi dikkate alınması gereken stratejik özellikler ortaya çıkar. Konsept geliştirme aşamalarında oyun atmosferinin uzay olmasına karar verildi ve oyuncu ve diğer nesneler basit geometriler olarak çalışıldı, bu nedenle oyunun adı "spaceBox" oldu. Kavramsal fikrin arkasındaki temel amaç, oyun algoritmasının minimalizmiyle paralel olarak basit ve hiçbir insan grubu üzerinde negatif çağrışımlarda bulunmayacak nötr bir kurguda oyunu ilerletmektir.

Sorumlu Yazar:
akdags22@itu.edu.tr

Akdağ, S. (2023). spaceBox Game. JCoDe: Journal of Computational Design

Anahtar Kelimeler: Calibri Light 9.5 punto, virgül ile ayrılmış, en fazla 5 anahtar kelime yazılmalıdır. Her kelime büyük harfle başlamalıdır.

1. Overview

An arcade game design and coding methodology is created to investigate the computational programming algorithms on game development as a final project submission to Istanbul Technical University, Department of Informatics, Program of Architectural Design Computing Msc., *Computer Programming in Architecture* class. This project is an investigation on how the famous arcade game called "Flappy Bird" can be interpreted to develop such a simple two-dimensional game. The programming language selected to develop the game is Processing, due to its user-friendly interface and easy visual and algorithmic testing possibility. The game provides a single play mode for its users to navigate the white box character so that it can move and gain points. During a single game experience, the users have one life to lose or continue moving forward. The main aim is not being touched by the obstacle objects called enemies in the code, as long as the player keeps the white rectangle away from the enemies, the score is rising up in the top left corner. An interface for the concept of space is designed on a two-dimensional layout in Adobe Photoshop. The space background image is implemented as the medium of the game environment in Processing software.

The tutorials related on how to code the famous arcade game are watched on YouTube as to source code forming training so that the algorithmic logic of such related examples can be understood and a similar yet game can be designed with its unique and original characteristics of its own. Processing software allowed creating a main code of game working, and some related additional classes are created and integrated inside the main code, one class for enemy object creations and definitions, one class code for the score creations. All the game when considered as one is strategically merged on a frame by frame action based on an algorithmic strategic foundation on Processing Software.

2. Interface Design

The user interface is developed upon the concept of creating a neutral connotation in terms of not discriminating anyone due to gender, country or any possible divergence in the life of people, as the methodology to define a target audience of players, and a set up of

space concept is found to be coherent to be attractive to all possible target players of the game. Therefore, first the pixel of the set up of the game is defined on Processing software as 1440 x 300 pixels. Then, using to create background and game objects, Adobe Photoshop is used. The background image is a collage of space elements like stars, and pixel clouds are generated and put in the visual to color the background. A simple path-like image is aimed so that, both the space vibe would be satisfied, and the game path player and the obstacle objects would be visible for the comfort of the player. The final outcome background image is as seen in **Figure 1**.

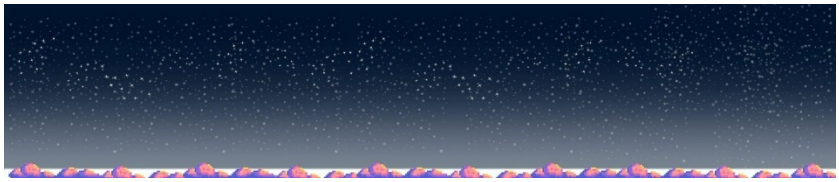

Figure 1: Background Image Design

In order to attain the characters as the spaceship and enemy bombs, pixel objects are created and saved as backgroundless png objects. The design with the objects and object moving directions are planned as **Figure 2** represents.

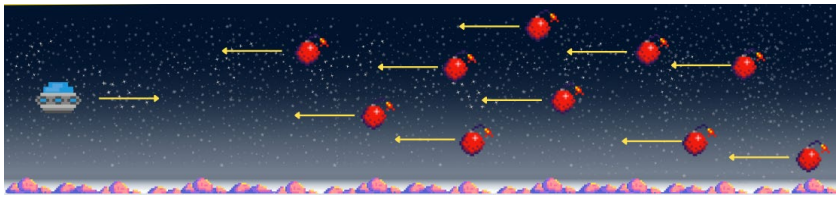

Figure 2: Interface Design with the Pixel Objects

An another main concern about the game, texts are planned to show the score and the playing the game button, a font is downloaded in the name of “ADAM” from online sources, and it has been added into the “data” folder of the project. The Play Button text and the score text on the top left corner is shown by **Figure 3** and **Figure 4** correspondingly.

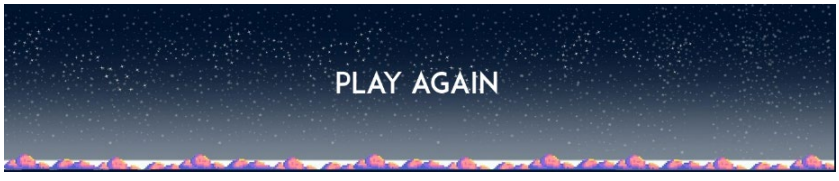

Figure 3: Play Button Text

Figure 4: Interface with the Score Board

All the design elements are collected in “data” folder and used during the Processing scripting of the game procedure. The article is going to be showing the algorithmic relationships of the design and the game development strategies in the upcoming sections.

3. Game Algorithm Scheme

In order to be able to start a coding work of the spaceBox game on the Processing software, a map of the planned algorithm is created as a flowchart, so that rules are set up, and the conversion of the game design into code can be done step by step and be represented in the Processing language. From the game start to the death of the player and “play again” option, the game algorithm scheme is formed as **Figure 5**.

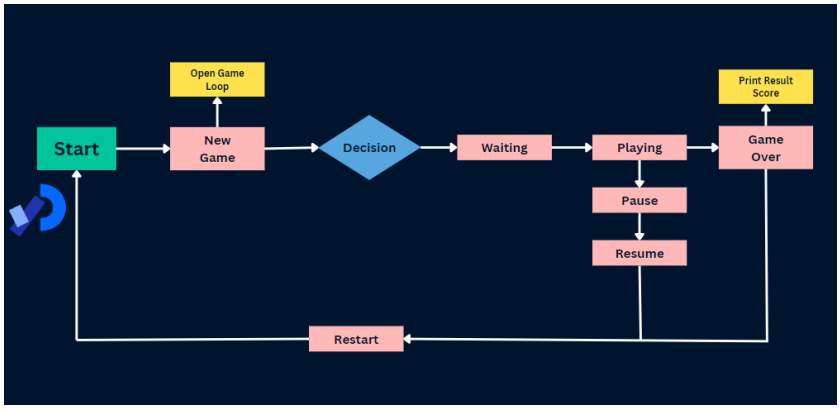

Figure 5: The Algorithm Scheme for spaceBox Game

4. Coding Procedure & Processing Software Work

In the definition section of Processing work before starting the set-up part, to create spaceBox game elements like images to be imported, player values of size, position, speed, acceleration, gravity effect, Boolean definitions for game control, font and enemy class array are all defined. The complete initial definitions for the game in the main code section is shown by **Figure 6** below.

```
//background image
PImage bg;
PImage ship;
int y;
// radius of player
int radius = 15;
// the horizontal speed
int horiz = 5;
// jump
int jump = 5;
// position of pl
float xp;
float yp;

// timer
int begin;

// speed of pl
float xs = 0;
float ys = 0;

// slowing down
float r = 0.7;
// gravity
float gravity = 0.4;

// the booleans of movement
boolean upPressed = false;
boolean downPressed = false;
boolean rightPressed = false;
boolean leftPressed = false;


int enemyNumber = 10;

boolean alive = false;

boolean won = false;
// the text
PFont font;
// the wave movment for the enemies
float os;
float period = 300;
float amplitude = 100;
// the array
enemy[] enemys;
```

Figure 6: Processing Interface Game Code Setup

The code continues with void setup part where functions are initiated to be drawn or worked on by the game. In the set-up part background image import, player box placement, enemy array of random 10 orange

boxes, related texts, buttons are all set up so that, operations on the set-up items created can proceed with the further steps of user experience on the game. The set-up codes are shown by Figure 7.

```
void setup() {
  // size of background
    size(1440, 300);
    bg = loadImage("background.jpg");

  // starting position of pl
  xp = width*0.2;
  yp = height/2;

  // enemy array
  enemys = new enemy[enemyNumber];

  // rectmode setup
  rectMode(CENTER);
  // text setup
  textAlign(CENTER, CENTER);

  noStroke();
  // import of font
  font = createFont("ADAM.otf", 100);
  textFont(font,50);

  for (int i = 0; i<enemys.length; i++) {
    enemys[i] = new enemy(-16, 12, 6);
  }

  startScore = new Score (0);

}
```

Figure 7: Processing InterfaceSet-up Codes

The draw part deals with the code to sustain the conditions to continue the game as alive player or die and have the condition of death. The possibilities are created by the if loop coding, as Figure 8 shows.

The control and movement of the player is characterized by the code with the operations in Figure 9. The user game playing items as the keys are functionalized so that movements and gravity and navigation initials are experienced and controlled by the user by attaining functions to key-pressed actions of the user. The code showing the key-pressed attainment is shown by Figure 10.

```
  if (alive) {
     //score
     fill(200);
     text("Score = ", 200,50,50);
     begin = millis()/1000;
     fill(200);
     text(begin, 370,50,50);

    // control the pl
    controll();

    move();

    walls();

    display();

    for (int i = 0; i<enemys.length; i++) {

      enemys[i].run();

  //startScore.countUp();
  //fill(200);
  //text(startScore.getTime(),370,50,50);
  //text(startScore.getTime(),370, 50,50);

    }

} else if (!alive) {

    if (!won) {

      fill(255);

      text("play again", width/2, height/2);

    } else if (won) {
      // then make the text green
      fill(0, 255, 0);

      text("YOU WON \n PLAY AGAIN", width/2, height/2);
    }
  }
}
```

Figure 8: Processing Interface If Loop Codes

```
136 void controll() {
137   // go to right
138   if (rightPressed) {
139     xs = horiz;
140   }
141   // go to left
142   if (leftPressed) {
143     xs = -horiz;
144   }
145   // go up
146   if (upPressed) {
147     ys = -jump;
148   }
149   // go down
150   if (downPressed) {
151     ys += jump/2;
152   }
153 }
154
155 void move() {
156   // speed + position
157   xp += xs;
158   //speed + position
159   yp += ys;
160
161   // gravity + speed
162   ys += gravity;
163   // speed + resistance
164   xs *= r;
165 }
166 // player pl
167 void display() {
168
169   fill(255);
170   rect(xp, yp, radius*2, radius*2);
171 }
172 // solid walls
173 void walls() {
174   // top solid wall
175   if (yp <= radius) {
176     yp = radius+1;
177     ys = -ys*(r/2);
178   }
```

Figure 9: Processing Interface Move and Control Operations

```
198 // booleans setup
199 void keyPressed() {
200
201   if (keyCode == UP) {
202     upPressed = true;
203   }
204
205   if (keyCode == DOWN) {
206     downPressed = true;
207   }
208
209   if (keyCode == LEFT) {
210     leftPressed = true;
211   }
212
213   if (keyCode == RIGHT) {
214     rightPressed = true;
215   }
216 }
217
218
219 void keyReleased() {
220
221   if (keyCode == UP) {
222     upPressed = false;
223   }
224
225   if (keyCode == DOWN) {
226     downPressed = false;
227   }
228
229   if (keyCode == LEFT) {
230     leftPressed = false;
231   }
232
233   if (keyCode == RIGHT) {
234     rightPressed = false;
235   }
236 }
```

Figure 10: Processing Interface Booleans Function Attainment to Keys

4.1. Classes

Object oriented design is integrated into the game design in order to differentiate some elements from the game to work in its own algorithm and the working elements as classes are in need in the game to be a complete arcade game model. First class created in the game is the enemy class to define how the randomly created 10 obstacle boxes are working and the properties to move, position on the screen

randomly and the ability to kill the player when in touch with them is given to the class by the code shown by **Figure 11**.

```
class enemy {
  // speed of enemy
  float speed;
  // x position of enemy
  float xl;
  // y position of enemy
  float yl;
  // radius of enemy;
  int rad;
  // row size;
  float rowNumber;
  // size of spawning rows;
  float rowSize;
  enemy(int spt, int radt, int rowNumbert) {
    speed = spt;
    rad = radt;
    rowNumber = rowNumbert;
    rowSize = height/rowNumber;
    // respawns the enemy
    respawn();
  }
  // makes the enemy move
  void move() {
    xl += speed+os;
  }
  // random position
  void respawn() {
    xl = (int)((int)(random(0, width/rowSize))*rowSize)+width;
    // xl = int(random(radius, width-radius));
    yl = (int)((int)(random(1, rowNumber+1))*rowSize-rowSize/2);
  }
  // kill the player
  void collide() {
    if (abs(xp-xl)<rad+radius && abs(yp-yl)<rad+radius) {
        alive = false;
    }
  }
  void teleport() {
    if (xl<-radius) {
      respawn();
    }
  }
  void display() {
    fill(255, 150, 0);
    rect(xl, yl, rad*2, rad*2);
  }
  void run() {
    move();
    teleport();
    display();
    collide();
  }
}
```

Figure 11: Processing Interface "enemy" Class

Another class need occurred when a score counter was in need to be created and implemented in the game coding procedure. First attempts to create the score counter by distance the player is achieved to make failed to give a gradually increasing score, so a timer is designed inside

the “score” class, and with speed adjustments according to the frame rate, an increasing score on the top left of the game interface is achieved. The class code is shown by Figure 12.

```
class Score
{
  float Time;

  Score(float set)
  {
    Time = set;
  }
  float getTime()
  {
   return(Time);
  }
  void setTime (float set)
  {
   Time = set;
  }
  void countUp()
  {
  //Time += (1/frameRate*0.001*0.95);
  Time += (1/frameRate);
  }
  void countDown()
  {
   Time -= (1/frameRate*0.001*0.95);
  }
}
```

Figure 12: Processing Interface “score” Class

5. Conclusion & Suggestions for Further Improvement

The whole project is completed as an arcade game design task at the end, and the resulting game worked well accept minor enhancements can be made in the further investigations to create a more fun and more complex game outcomes. The feeling of movement, speed, gravity, and the desire to play the game again and again were the ultimate successes of the whole work. The design of space environment was also successful however the characters of spaceship and enemy bombs could not be integrated in the game. For further improvement, dual play options can be developed, keeping the track of score for the top 10 scores could be achieved, level creation can be fun for the players to experience different backgrounds, enemies, and harder speed conditions. Music as a very effective tool for many games could

be integrated into the game and opening and closing the music could be a button option at the beginning of the game. Many alternatives to these improvement suggestions could be proposed to enhance the joy experience of the game by its players, however with its simplicity, minimal and nostalgic two-dimensional pixel design interface, and attracting the players to play the game again for many times, the *spaceBox* game succeeded standing out as a well-working optimal arcade game for its users.

6. Related Links

Processing Game Code & Interface Folders:

https://drive.google.com/drive/folders/16GUv5NMdr2YYSKo9lEiFi6te138w4nIa?usp=sharing

Game Video Link:

https://drive.google.com/drive/folders/1pcQNJnYCpmHZTiJMaLfgCppGzJ3389Q8?usp=share_link

Presentation:

https://drive.google.com/drive/folders/1TMba2_UnGz3801IZLNuIwWEKd1duJXkm?usp=sharing

References

Books

Harbour, J. S. (2007). Game programming all in one. Thomson Course Technology.

Web Resources

Facredyn, A. (2017, April 2). Processing game tutorial 1. YouTube. Retrieved January 21, 2023, from https://www.youtube.co Facredyn, A. (2017, April 2). Processing game tutorial 1. YouTube. Retrieved January 21, 2023, https://www.youtube.com/watch?v=QJGcxeF3YBs&list=PLZDenWpA-ZLxr8r_UX8XpWc-RXNG4Foez&index=4&t=233sm/watch?v=QJGcxeF3YBs&list=PLZDenWpA-ZLxr8r_UX8XpWc-RXNG4Foez&index=4&t=233s

Journals

Santos, S. S. (2021). The science behind Flappy Bird. 2021 IEEE Integrated STEM Education Conference (ISEC). https://doi.org/10.1109/isec52395.2021.9763893