9th International scientific conference
Technics and Informatics in Education – TIE 2022
16-18 September 2022

Session: *Educational Technology*
Professional paper
DOI: 10.46793/TIE22.088P

Android game development

Selver Pepić [1*], Marija Mojsilović [1], Muzafer Saračević [2]
[1] Academy of Vocational Studies Šumadija, Department of Trstenik, Trstenik, Serbia
[2] University in Novi Pazar, Department of Computer Science, Novi Pazar, Serbia
[*] spepic@asss.edu.rs

Abstract: Mobile devices that are today at the global level with the largest usage rates are precisely those devices that are based on the Android OS platform. The work was based on the description of the created project, that is, the android game. Android games are the areas that are most represented in the IT development world, that is, the world of the mobile development application. Here, the focus is on that part of the application software, which is an integral part of the entire android package. The android system itself provides the opportunity as an open platform programming application solutions and install them on a mobile device. Through the work, its content and code in the form of scripts will be explained.

Keywords: *Android; Development; C#; Game*

1. INTRODUCTION

As everyone knows today, the mobile industry is expanding, mobile devices are those that have almost completely replaced computers today, that is, they found such wide use, that it is almost impossible to imagine life without smartphones, that is, how many of them are called "smart" devices. Mobile devices that are today at the global level in the largest usage rates are precisely those devices that are based on the Android OS platform. Every operating system, even the one on mobile devices, is composed of its operating core layout to the software part visible to the user, is, application and system software.

Here, the focus is on that part of the application software, which is an integral part of the entire android package. The android system itself provides the opportunity as an open platform programming application solutions and install them on a mobile device. That's exactly what it is

the subject of this paper, that is, an android game, which, as we know, is something without which one

an average user cannot imagine using their android device, that is, a smartphone [1].

2. ANDROID OPERATING SYSTEM

Android as a type of mobile operating system is a type of platform (OS) which open source, it is designed for creating mobile applications for other devices that starts the operating system [2].

It is very important to note that the android operating system consists of "native" applications, that is, in addition to them, from applications developed by others for free (free) installation, that is, download, or option with purchase (buy). Native android applications are pre-implemented in the android operating system, that is, they are native, as the word itself suggests, applications without which the Android operating system can not function, that is, it would not provide users with a complete experience.

For example, Java will be used to develop an Android application on one of the Java platforms DevKit. For iOS, the iOS SDK will be used, while some types of Windows platforms will use .NET framework SDK.

Android is therefore an operating system that is not used by smartphones, but also by devices such as tablet devices. It originated from an American company that is globally known, and in It represents a giant in the IT world, and that is the company Google Inc. This type of OS is based on the core of one of the most well-known OS Linux, and we can recognize it in its composition of open source software [2].

Android OS has gone through many changes in its rich history. Its public presented in a very large number of versions, some of which crashed, and some of which did not receive support, while some are listed that are still used today and receive support. The following are the Android OS versions:

- Asteroid OS (v 1. 0)
- Petit Four (v 1. 1)
- Cupcake (v 1. 5)
- Donut (v 1. 6)
- Eclair (v 2. 0-2. 1)
- Froyo (v 2. 2-2. 2. 3)
- Gingerbread (v2. 3-2. 3. 7)
- Honeycomb (v 3. 0-3. 2. 6)
- Ice Cream Sandwich (v 4. 0-4. 0. 4)
- Jelly Bean (v 4. 1-4. 3. 1)
- KitKat (v 4. 4-4. 4. 4)
- Lollipop (v 5. 0-5. 1. 1)

- Marshmallow (v 6. 0-6. 0. 1)
- Nougat (v 7. 0-7. 1. 2)
- Oreo (v 8. 0-8. 1)
- Pie (v 9. 0)
- Q (v 10. 0)
- Red Velvet Cake 11.0
- Snow Cone 12.0

2.1. Basic methods in the application life cycle

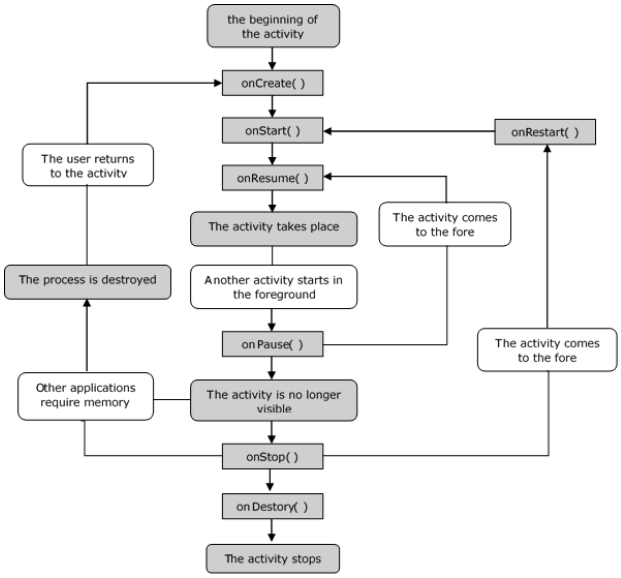

Figure 1. *Presentation of life cycle methods [3]*

Fig. 1. shows a graphical representation of the life cycle methods of Android applications.

onCreate() method is an "introductory" method to the application, it can be run when activating the application by the user, when starting this method, next the method that can be seen in the attached image is the onStart() method, which is not yet visible, until the onCreate method mentioned above is fully executed.

The **onStart()** method represents the method that will start its execution after completely executing the onCreate() method, that is, her UI will be visible right then.

onPause() the method can be activated by pressing a certain part of our interface applications, this method will be executed if the user pauses the action, or if it is current force stop, and if you completely leave the application after it, it will be activated onStop() method, but otherwise, the method that will wait for execution is the method onResume().

onResume() method is just fired after the user activation of the onPause() method, it is waiting for approval for its execution by the user, after pausing it.

The **onStop()** method represents the method that results from the activation of the onPause() method, that is, if the user does not decide to return to the application interface, that is turn off all background processes when pausing it, this method takes effect and completely shuts down the application and activates the last method in the application cycle, which is the onDestroy() method.

onDestroy() method is the one preceded by the activation of the onStop() method, it is credited to "kill" the application [3].

3. ANDROID GAMES

This type of software solution is considered an indispensable type of everyday use smart devices, primarily among the younger population. In the field of android games, there are games with educational and entertaining characters, and as such, they represent an indispensable part of the Android OS application software.

Today we see that games have greatly expanded to smart devices, and even some desktop application solutions, are also being made on mobile devices, with serious optimization. Knowing all that, we must also know that today in the IT world an incredible concentration of programmers is involved in web/android game development, that is, by developing games for certain platforms, that is, with the help of certain programs language.

In the following, the work is based on the description of technologies that are used, as well as a description of the project software used to create an android game. Technologies that are very present in the world of android games will be described in more detail, technologies that were used, namely: Unity, C#, and Visual Studio Code editor served for typing and editing the complete code that was used in the creation of the game that I will present as my project solution. The process of its installation will be presented, and some interesting details about the preparations of the aforementioned Unity IDE, that is, a powerful software package that is a coupling between the C# programming language and the API.

4. Technologies in the development of project solutions

The necessary tools with which the project solution is completed are the tools that are used today to use those innovative solutions to complete android games. A programming language was used C#, and the implementation in the code editor Visual Studio Code, and everything is included in the development environment called Unity, which works in conjunction with the aforementioned programming language. The mentioned technologies will be described below, how they function, and in the best possible way illustrate the process of creating an android game.

4.1. Description of the C# programming language

C# is one of the languages that is one of the youngest in the palette of all accessible programming languages that we can study and improve. Namely, this language was created as a product of Microsoft's development environment in 2000 [4].

We must mention that it is in the development team was Andres Heilsberg, which is also one of the most deserving for his commercial use.

This programming language supports multiple paradigms and supports the following paradigms: object-oriented, imperative, declarative, and generic, like most modern ones programming languages, such as Java and C++. Language is a general course of change and is the most suitable for the development of applications for the .Net Framework platform [4].

Today it is in conjunction with the IDE Unity package and used as a combination, they serve to create android applications, which can be presented and created as a 2D and 3D version of the game, it enables the powerful Unity, which will be discussed later in the paper [5].

5. Visual Studio Code

Visual Studio Code is a lightweight yet powerful source code editor that runs on a desktop and is available for Windows, macOS, and Linux. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity) [6].

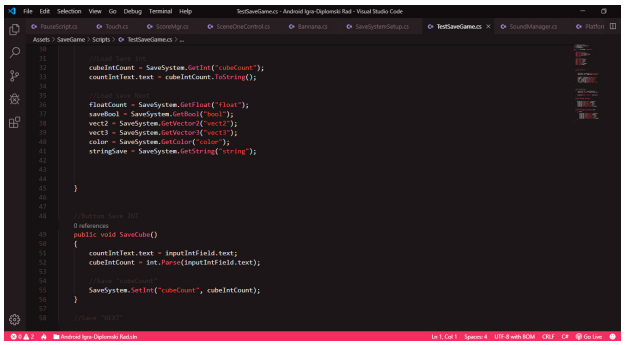

Figure 2. *Single window display in the VS Code editor*

In Figure 2, above, you can see a graphical representation of the Visual Studio Code environment editor.

It is important to note that Visual Studio Code supports the so-called extensions, which enable automatic completion of the first line of code through suggestions from extensions, it was done in such a way as to make the work of developers as easy as possible, using of extensions which are additionally implemented in a very simple way and incorporated into the autocomplete editor function, which will solve excessive syntax problems, and that they will be typed into a record soon.

5.1. Creating a Unity desktop

Unity represents an environment through which we can get a palette, that is, a variety the possibility of creating mobile applications, the creation process will be shown in the following example new project, and how to choose a 2D or 3D platform for creating an Android game.

1. Step

This step is shown in Figure 19, whereby pressing the "New" button we create a new project and that's how we start creating our project solution. After this step, I will illustrate finalizing the creation of the project, and later work will be based on a detailed description of the application which is calculated in the mentioned software.

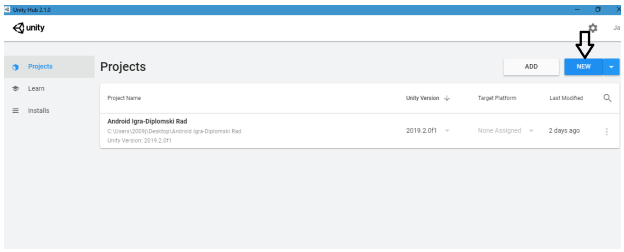

Figure 3. *Display window for creating a new project*

2. Step

After pressing the "NEW" button, we will get a window with additional settings for our project, where we will be able to enter the name of the project, its location, the place where it will save the project solution, and the platform, that is the work template. We can choose a 2D template, which will be for creating 2D applications, which is also implemented in my project solution, you also have a choice of 3D templates, and 3D with add-ons.

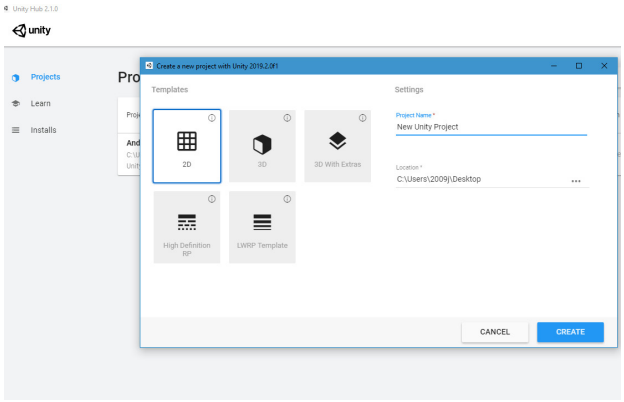

Figure 4. *Presentation of the template selection for the project solution*

As shown in Figure 5 graphic representation of the project desktop.

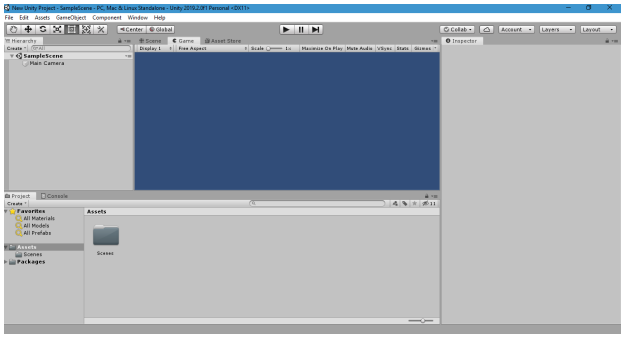

Figure 5. *Project Desktop*

6. PROJECT SOLUTION DETAILED ANALYSIS

6.1. Project solution using programming language C# and Unity + VS Code

An Android game that has been completed is a game that has been built on the previous one to the clarified template containing Unity, it was made in the 2D template and with help of the programming language specified, all objects and characters are manipulated and implemented in the project solution through Unity. Android game, that is project solution which was completed consisted of a series of implemented objects, that is, characters via assets in Unity, object, that is, characters are pre-designed and represent characters that are or drawn, or created through software tools designed for the same [7].

The game has a fun character, that is, it represents a kind of casual variant of android games, where you start your main character, a little monkey, by touching your display, on a smart device, where your goal is to collect as many objects as possible in your view banana, which will bring you good results, such as avoiding contact with an enemy object.

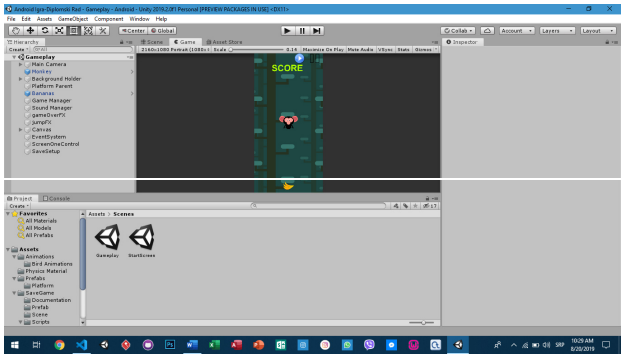

Figure 6. *The initial view of the workspace and layout of the 2D game scene*

As you can see in Figure 6, the initial screen, or layout, is shown applications, you have already learned about the working environment in the previous parts of the work. On the left side of the window, you can see that that part is reserved for the space of all Canvas, storage audio files, and all other objects present in the game.

In the particular picture, it is activated scene 1 which is named "Gameplay", and the same scene contains the initial window, that is, everything that you see on it when starting the game is related to the "StartScreen" scene.

After starting, the scene from the picture that you have the opportunity to see visually is automatically started. In the lower part of the Unity workspace, you see a space reserved for accessing the desired objects, scenes, and all other elements inserted into the environment, which is the whole process the makes the development of the game easier, that is, the accessibility of the elements is great. Implementation is on in the English language because I find it much easier to work with acronyms from their spoken language areas.

6.2. Individual description of all available folders in the project solution

In the presented folder, you can graphically see in Figure 6. the characters that are implemented in the project solution. That folder contains the main character, the little one, in its bundle monkey implemented in the environment called "Monkey", he is the main character which we manipulate through the programming language C#, while the bird and bananas are of course objects that perform interaction with the main character to achieve fun and a logical whole.

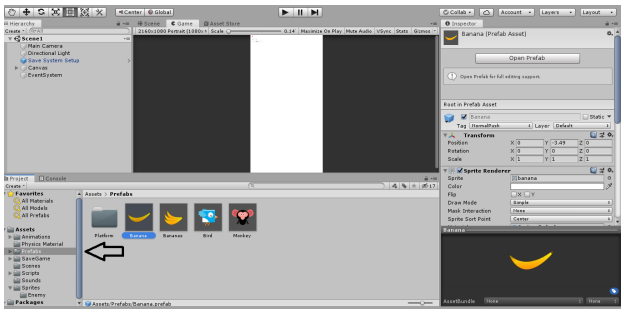

Figure 6. *View Prefabs folder*

In the folder "prefabs" we also have a folder with platforms, which contains objects, i.e. how popular are called assets, which are also used in this project solution for interaction with given objects. The folder itself is a space that is organized for storing the object material used in the project solution.

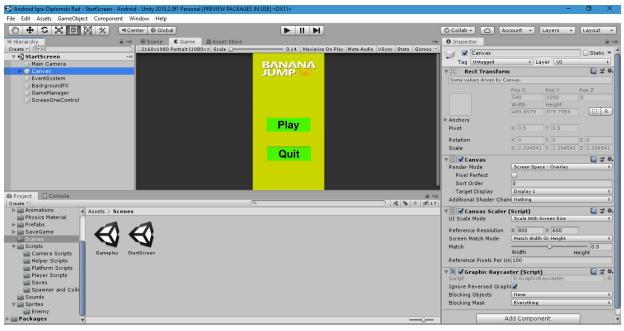

Figure 7. *Display the opening scene*

6.3. Scene 1 StartScreen

Figure 7 shows the appearance of the initial screen of the first scene of the android game. Like what you see in the left bar are all folders with manipulative scripts in their possession, that is the scripts that regulate the work of this initial scene 1.

The scene itself is set to a certain resolution, which is the appropriate resolution for most smartphones. It is in portrait with an aspect ratio of 2160x1080px. You can set yours right next to the "Scale" slider in the upper part of the workspace bar, under the name "Free Aspect".

On the right side of the window, canvas scene 1 of our android game is shown, and there are several tools on it are deserving for that design part, ie the part where you can adjust the position of this scene on your screen, whereas in my listening you have displayed values for the scene in the given image. The canvas folder and the main camera are responsible for the positioning, that is, the design of the entire scene, in the Canvas folder contains a series of elements that are shown in scene 1, that is the "Play" button, and the button "Quit", with the background color implemented in this scene, and the logo on the top bar scenes.

The work will further be based on the implementation of some code scripts, used for manipulation with objects from the Canvas, that is, the entire scene 1, the scene called "StartScreen".

6.4. Touch Script character gesture

A script to regulate fluidity and enable the operation of the entire game on the touch display of the smart device.

```
1   public class Touch : MonoBehaviour {
2     public GameObject player;
3     public int speed;
4     public float smoothing;
5     public bool pause;
6
7     void Update () {
8       if (EventSystem. current. IsPointerOverGameObject() ||
9         EventSystem. current. currentSelectedGameObject != null)
10      {
11        return;
12      }
13      if (Input. touchCount > 0 && Input. GetTouch(0). phase ==
14  TouchPhase. Stationary && pause == false)
15        {
16          Vector2 touchPosition = Input. GetTouch(0). position;
17          double halfScreen = Screen. width / smoothing;
18
19          //Check if it is left or right?
20          if (touchPosition. x < halfScreen)
21          {
22            player. transform. Translate(Vector3. left * speed *
23  Time. deltaTime);
24          }
25          else if (touchPosition. x > halfScreen)
26          {
27            player. transform. Translate(Vector3. right * speed *
28  Time. deltaTime);
29          }
30        }
31    }
32  }
```

The script shows in its first part which objects it interacts with, in the initial part of the code implemented the class "Touch" which is of type "MonoBehaviour". MonoBehaviour is the base class from which every Unity script derives.

When using C#, you must explicitly derive from MonoBehaviour. The following code includes instances of the class which you have the opportunity to see inside the "Touch" class, and the Update() method, which doesn't have one return result because it is of type Void.

The method itself takes the vectors, that is, the position of the main character, and if the game user presses the right or the left half of the X axis, then the object, that is, the main character of the Android game will move in the direction of the pressed part of the screen, that is, the left or right half, where the direct speed of character movement will depend on the predefined character speed value that will be the case to multiply with vectors whose values are defined in the script of our character.

```
1   public class SceneMgr : MonoBehaviour
2   {
3     public void StartGame() {
4       Application. LoadLevel("Gameplay");
5     }
6     public void Quit() {
7       Application. Quit();
8     }
9   }
```

The script you have the opportunity to see shows how the solution for launching the second stage of our project solution. We have a script class of the same name in the shown code, which in its algorithm contains the StartGame function that activates scene 2 "Gameplay", and if the user decides to leave the application, then the Quit function will be in action activated and will completely stop background processes and the application. An audio file, in mp3 format, was implemented in the initial scene for audio coverage of the scene, it's arbitrary, so you can implement any mp3 file, it's up to the game developer.

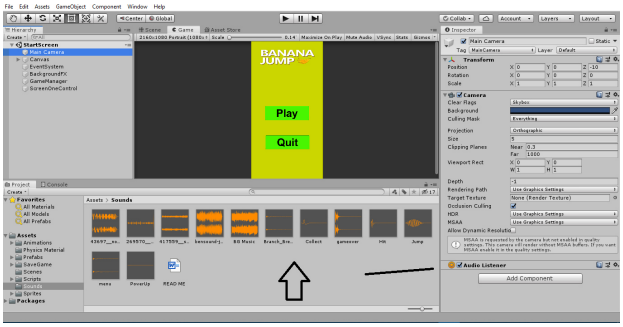

Figure 8. *Audio files and implementation*

On the stage you can find two buttons "Play" and "Quit", both buttons have implemented the function, which is predefined onClick(), through which they trigger, with scene 2 or with the

onDestroy method, if the Quit button is pressed. The following image illustrates the solution in Unity environment.

6.5. Script for creating events and buttons

```
public class Button : Selectable,  IPointerClickHandler,
ISubmitHandler
    {
        [Serializable]
        public class ButtonClickedEvent : UnityEvent {}
        [FormerlySerializedAs("onClick")]
        [SerializeField]
        private ButtonClickedEvent m  OnClick = new ButtonClickedEvent();
        protected Button()
        {}
public ButtonClickedEvent onClick
        {
            get { return m_OnClick; }
            set { m  OnClick = value; }
        }

        private void Press()
        {
            if (!IsActive() || !IsInteractable())
                return;

            UISystemProfilerApi. AddMarker("Button. onClick",  this);
            m_OnClick. Invoke();
        }
public virtual void OnPointerClick(PointerEventData eventData)
        {
            if (eventData. button != PointerEventData. InputButton. Left)
                return;

            Press();
        }

public virtual void OnSubmit(BaseEventData eventData)

{
            Press();
            if (!IsActive() || !IsInteractable())
                return;

        DoStateTransition(SelectionState. Pressed,  false);
        StartCoroutine(OnFinishSubmit());
    }

    private IEnumerator OnFinishSubmit()
    {
        var fadeTime = colors. fadeDuration;
        var elapsedTime = 0f;

        while (elapsedTime < fadeTime)
{
            elapsedTime += Time. unscaledDeltaTime;
            yield return null;
        }

        DoStateTransition(currentSelectionState,  false);
    }
  }
}
```

6.6. Scene 2 Gameplay

The scene named Gameplay represents the second scene in the given project solution. This scene is exactly that "toy" scene, where we can see the whole way that everything functions as a whole.

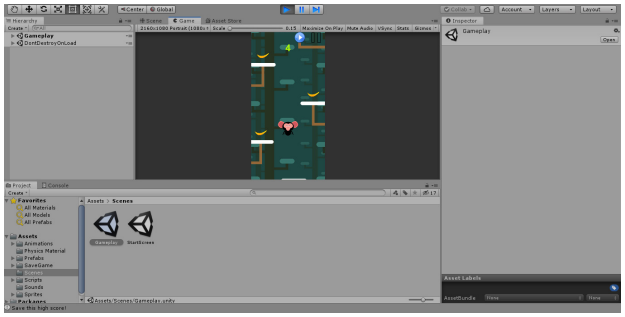

Figure 9. *Gameplay scene layout*

Figure 9 shows the Gameplay scene and its Unity environment. Of course, as I mentioned this scene is triggered when you click the "Play" button from the previous scene which is called "StartScreen".

The scene itself is in our "Gama Manager" object, inserted into the Unity environment, which you can see in the structure shown in Figure 10.

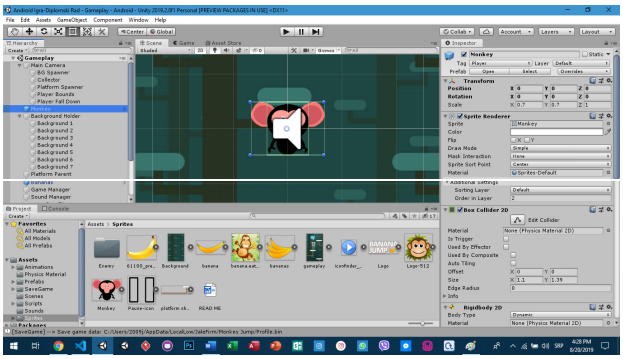

Figure 10. *Gameplay scene elements*

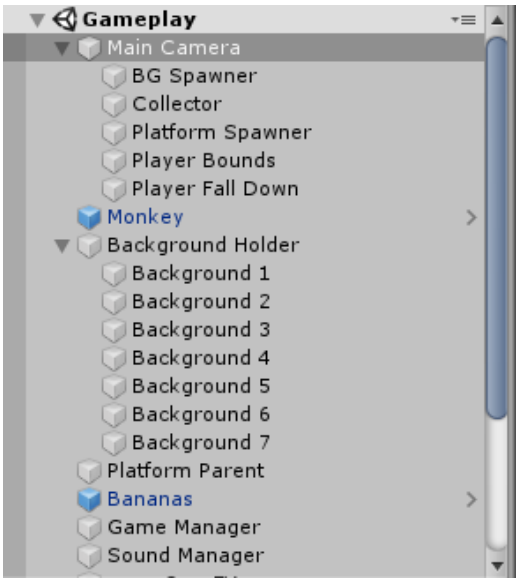

Figure 11. *Graphical representation of the Gameplay scene folder*

6.7. Main character script and implementation

As you can see in the attached image, the character of the Android game is located in folder assets, and the folder inside it is named Sprites. This character is implemented in the Unity environment into scene 2 as the main object. On the side in the bar to change the value, change positions contain clearly defined positions along the X and Y, and Z axes. Contains Box Collider which is in 2D mode and through which he will interact with his environment. It also contains in its composition Rigidbody, also intended for 2D games, in which we freed up the Z-axis rotation because it is it meant that he would not rotate unnaturally during the jump, but along the X and Y axes.

```
1   public class PlayerScript : MonoBehaviour {
3      private Rigidbody2D myBody;
4      public float move_Speed = 2f;
5
8      public float normal  Push = 10f;
9
10     public float extra_Push = 14f;
11
12     private bool initial  Push;
13
14     private int push_Count;
15
16     private bool player_Died;
17
18     public Text score;
19     public Text topScore;
20     public int point;
21     public int topEarnedScore;
22     public SaveGame saves;
23    public AudioSource gameOver;
24    public AudioSource jump;
25     void Awake() {
26        myBody = GetComponent<Rigidbody2D>();
27    }
28    private void Update()
29    {
30       topEarnedScore = point;
31       if (point > saves. earnedScore)
32       {
33          print("Save this high score!");
34          saves. earnedScore = topEarnedScore;
35          saves. SaveData();
36       }
37    topScore. text = saves. earnedScore. ToString();
38     }
39
40     void FixedUpdate() {
41        Move();
42        score. text = point. ToString();
43     }
44     void Move () {
45
46        if (player_Died)
47           return;
48        if(Input. GetAxisRaw("Horizontal")> 0) {
49
50           myBody. velocity = new Vector2(move_Speed, myBody. velocity.
51 y);
52
53        }else if(Input. GetAxisRaw("Horizontal")<0){
54
55           myBody. velocity=new Vector2(-move  Speed, myBody. velocity.
56 y);
57
58        }
59 }

60     void OnTriggerEnter2D(Collider2D target) {
61
62         if (player_Died)
63            return;
64
65        if(target. tag == "ExtraPush") {
66           point += 3;
67           if(!initial_Push) {
68
69              initial_Push = true;
70
71              myBody. velocity = new Vector2(myBody. velocity. x,  18f);
72
73              target. gameObject. SetActive(false);
74
75              return;
76    push;
77
78           }
79        }
80        if(target. tag == "NormalPush"){
81
82           point += 1;
83           myBody. velocity = new Vector2(myBody. velocity. x,
84    normal  Push);
85           target. gameObject. SetActive(false);
86           push_Count++;
87           JumpFX();
88     }
89           if(target. tag == "ExtraPush"){
90
91           myBody. velocity = new Vector2(myBody. velocity. x,
92    extra_Push);
93           target. gameObject. SetActive(false);
94           push  Count++;
95           JumpFX();
96        }
97
98        if(push  Count==2) {
99           push_Count = 0;
100          PlatformSpawner. instance. SpawnPlatforms();
101       }
102
103       if (target. tag == "FallDown" || target. tag == "Bird") {
104    player  Died = true;
105          GameOverSound();
106          GameManager. instance. RestartGame();
107       }
108    }
```

```
109    public void GameOverSound()
110    {
111       gameOver. Play();
112    }
113
114    public void JumpFX()
115    {
116       jump. Play();
117    }
118 } // class
```

The player script itself is a script that was created to manipulate the main character of my android game, this script is responsible for the complete logical interaction with the environment of the main character of the game, and the work of this script will be explained in more detail in the paper. Namely, the script contains the "PlayerScript" class of the same name, which has two functions in its composition Start() and Update().

At the end of the script we have an if loop where we enter parameters if our character comes into contact with the bird, that is, if it collides with it, then the character will die and we will activate the restart of the scene. Also at the end of the script, we have an inserted part for the audio coverage of the character death function, where it will when colliding with a bird, hear a certain arbitrary sound.

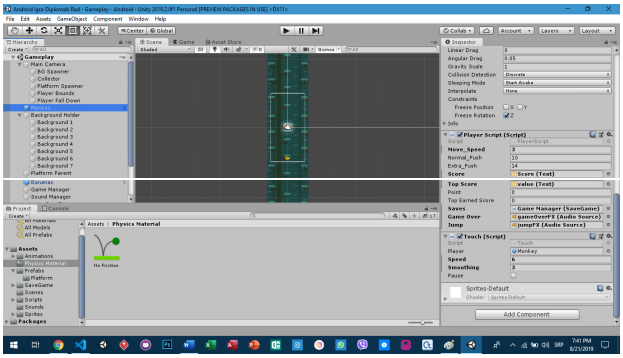

Figure 12. *Our character object and its script*

7. FUTURE WORK

The game is an antidote to boredom that is suitable for everyone. Games also have other benefits such as the brain development, solving problems, improving concentration, train speed, etc.

The mobile gaming market is an immeasurable, overall view of the mobile gaming industry. Games obviously also have more advantages, such as mobility advantages, they have more potential customers.

Considering that and the very popularity of the games, the result of this work and the development of the game presented in the work, we plan to place it on Google Play.

8. CONCLUSION

The work was based on the description of the created project, that is, the android game. Android games are the areas that are most represented in the IT development world, that is, the world of the mobile development application. This area requires a wide knowledge of many object-oriented

programming languages, and many development environments, and is considered one of the most demanding areas in the IT world. It is an interesting fact that today's programming languages in expansion precisely those that are compatible with development environments, as precisely the language that I processed in this paper, that is, which I used for the development of my project solution, which is C# [7].

This programming language, development environment Unity is a duo that is in the modern the app development world is at the top when it comes to Android games, the thought that a development environment capable of providing the user with the development of both 2D and 3D games is incredible, that is, it tells us enough about the demands and complexity of the given environment.

The paper shows in detail the preparation of software for current work. The project solution is graphically explained in detail through the scripts and images that illustrate the situation in the development organization. The user is enabled to completely create an image of what looks like an Android game development process. Complexity and demandingness are qualities that describe the process of creating an Android game, but it is safe to say that this area is one of the most enticing and beautiful in the developing mobile world, which he showed in his work specific example.

REFERENCES

[1] *James, T., Justin, M., (2014), Programiranje android aplikacija, Računarski fakultet*

[2] *Chinetha, K., Daphney Joann, J., Shalini, A., (*February 2015*), An Evolution of Android Operating System and Its Version,* International Journal of Engineering and Applied Sciences (IJEAS) ISSN: 2394-3661, Volume-2, Issue-2

[3] *Vikas, S., Manu, V., Pradeep, S., (February 2021), A survey of android application and malware hardening, Computer Science Review 39 (2021) 100365*

[4] *Jamie, C., (2015), Learn C# in One Day and Learn It Well, LCF Publishing*

[5] *Jayme, S., Brian, B., (2013), AndEngine for Android Game Development Cookbook, Birmingham*

[6] *Enrique Lopez, M., Diego, M., (2016), Android High-Performance Programming, Birmingham*

[7] *Miguel, C., Maria, G., Joaquim, E., Manuel, R., (December 2019), Using Android Tablets to develop handwriting skills: A case study, Heliyon 5 (2019) e02970*