Beykoz University

Department of Computer Engineering

Engineering Project 1.

“2D Platform Game”

Python

Final Report

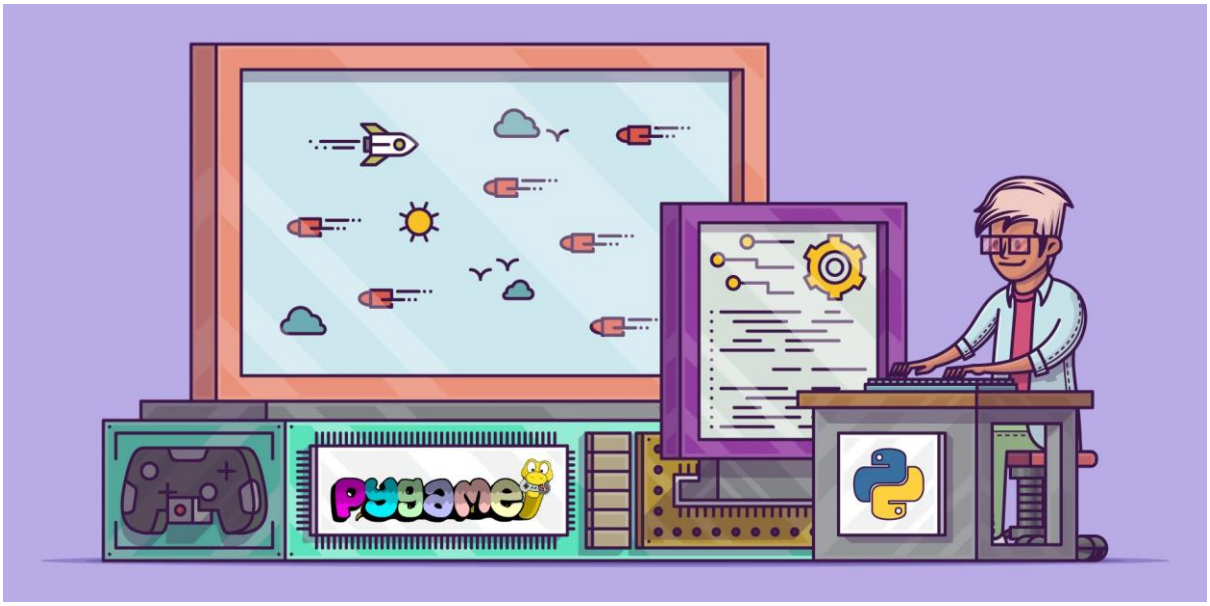
Leyla Abdullayeva-1904010038

Date of submission: 29 May 2020.

Table Of Contents

Introduction (Project Aim)

As you can see my project's name is "2D platform game" created by Python programming language.I've tried to do my best and show you the product that I'm working for 3 months.You can write whole games in Python using PyGame. Game Programming with Python is about building games using Python. It deals with general concepts of game development and specifics that apply when using Python for game development. Some of the general topics include simulations, game architectures, graphics, networking, and user interfaces. In this project we'll look at how we can use Python to create a 2D game. To solve this we would use PyGame which makes it easy to both understand the very basics of game programming and python. At the end of this project we will have a playable breakout game that we've written from start to stop using Python together with PyGame.



Information about Python

Python is an interpreted, high-level, general-purpose programming language. I used Python 3.8 programming language. Python is an outstanding language for people learning to program. Like music and movies, video games are rapidly becoming an integral part of our lives. Lately we've been spending a lot of time thinking about a game idea of our own, or exploring the possibility of making a career of this vibrant and growing industry.

Pygame and Arcade library

But where should we begin? Beginning Game Development with Python and Pygame is written with the budding game developer in mind, introducing games development through the Python programming language and the popular Pygame games development library. Of course, I haven’t learned “completely” how to create advanced games by taking advantage of the popular open source Python programming language and Pygame games development library. But I learned a lot about coding gaming preferences, sound, visual effects, and joystick/keyboard interaction. I’d investigated and discovered the concepts that are crucial to success in today's gaming industry, such as support for multiple platforms, and granting users the ability to extend and customize their own games.

In order to create a 2D game, we need to know a few things. One of them have to be the programming language that we are using. And the another important moment have to be the library. I used Arcade library from Python Game Development. Arcade is an easy-to-learn Python library for creating 2D video games. It is ideal for beginning programmers, or programmers who want to create 2D games without learning a complex framework.

IDE information (PyCharm)

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license. Pycharm is just an editor, you can use pretty much any editor, Pycharm is one of the best ones for Python out there, but might be a bit too advanced for a beginner.

Python has a lot of gaming libraries, just to name one, Cocos 2D, great for making simpler games like Super Mario Bros and Candy Crush.I used PyCharm 2019.3.3 version.

Prototype 1

Here, I want to talk about our Prototype-1.We did it as a group.I think me,Alperen and Yousef did great.While preparing our first prototype,we didn't really know the entire arcade library functions and etc.I'll explain you the exact steps and achievements.

One of the important points here is to install exact right thing from right and original address.Because if we'll install different and unoriginal program or IDE,it won't work and also it can damage our PC.

Installation of Python and PyCharm

Install Python from the official Python website:

https://www.python.org/downloads/

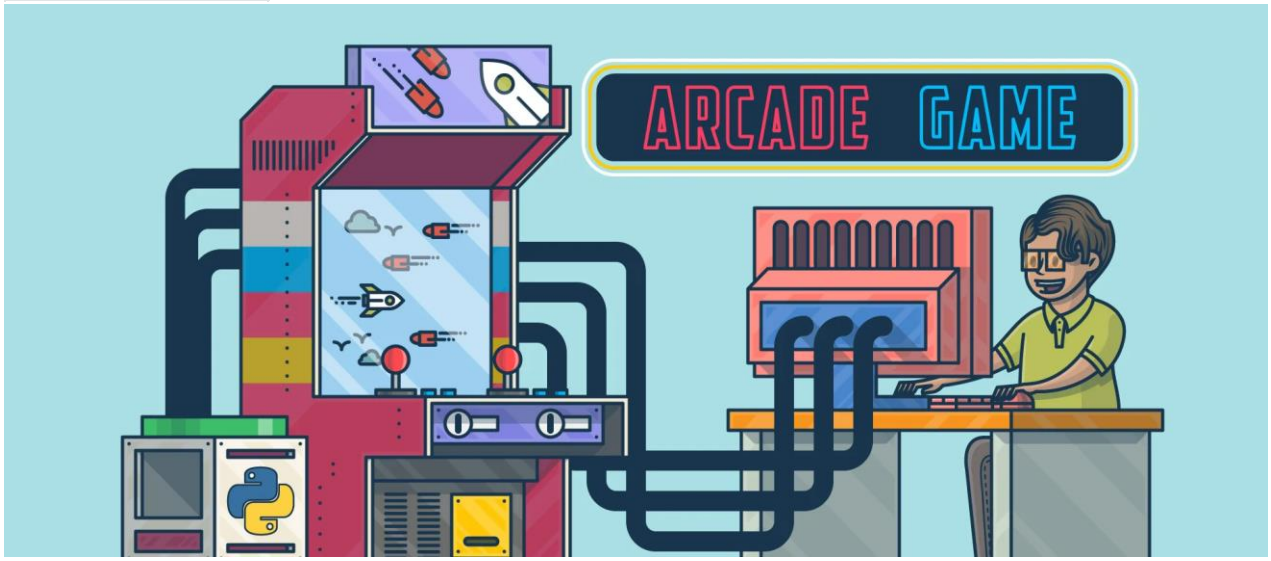
Installation of PyCharm (IDE)

https://www.jetbrains.com/pycharm/download/#section=windows

Installation of "Arcade Library"

Then we have to open our IDE and install the "Arcade Library". The best way is to use a "virtual environment." But I prefer to install this library via command line interface (CLI).Then I can install arcade directly using pip:

```
pip3 install arcade
```



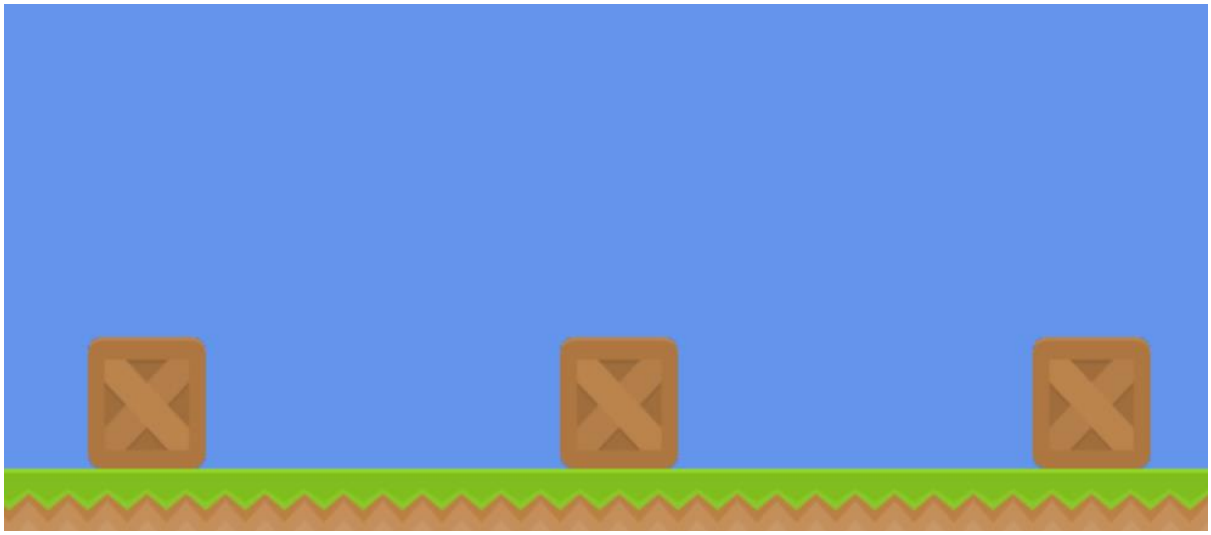
Opening a “Game Window”

In order to open a game window, you must’ve downloaded arcade library beforehand. As you can see, in here the game don’t have an actual name, so in the name part we wrote “Platformer”.

```
import arcade
import os

# Constants
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
SCREEN_TITLE = "Patformer"

# Constants used to scale our sprites from their original size
CHARACTER_SCALING = 0.8
TILE_SCALING = 0.5
COIN_SCALING = 0.5
SPRITE_PIXEL_SIZE = 128
GRID_PIXEL_SIZE = (SPRITE_PIXEL_SIZE * TILE_SCALING)

```

Sprite addition

After opening a new game window, the next step is to put in some graphics. “Sprites” are the graphical items that you can interact with. We create a sprite by using “Sprite” class.



User Control addition

We need to be able to get the user to move around.

First, at the top of the program add a constant that controls how many pixels per update our character travels:

```
# Player speed
PLAYER_MOVEMENT_SPEED = 5
GRAVITY = 1
PLAYER_JUMP_SPEED = 20
```

```
def on_key_press(self, key, modifiers):
    if key == arcade.key.UP or key == arcade.key.W:
        self.up_pressed = True
    elif key == arcade.key.DOWN or key == arcade.key.S:
        self.down_pressed = True
    elif key == arcade.key.LEFT or key == arcade.key.A:
        self.left_pressed = True
    elif key == arcade.key.RIGHT or key == arcade.key.D:
        self.right_pressed = True

    self.process_keychange()

def on_key_release(self, key, modifiers):

    if key == arcade.key.UP or key == arcade.key.W:
        self.up_pressed = False
        self.jump_needs_reset = False
    elif key == arcade.key.DOWN or key == arcade.key.S:
        self.down_pressed = False
    elif key == arcade.key.LEFT or key == arcade.key.A:
        self.left_pressed = False
    elif key == arcade.key.RIGHT or key == arcade.key.D:
        self.right_pressed = False
```

```
def on_update(self, delta_time):
    # We're calling physics engine

    self.physics_engine.update()

    # Update animations
    if self.physics_engine.can_jump():
        self.player_sprite.can_jump = False
    else:
        self.player_sprite.can_jump = True

    if self.physics_engine.is_on_ladder() and not self.physics_engine.can_jump():
        self.player_sprite.is_on_ladder = True
        self.process_keychange()
    else:
        self.player_sprite.is_on_ladder = False
        self.process_keychange()
```

Gravity and Scrolling addition

Gravity:

```
    def on_key_press(self, key, modifiers):
        """Called whenever a key is pressed. """

        if key == arcade.key.UP or key == arcade.key.W:
            if self.physics_engine.can_jump():
                self.player_sprite.change_y = PLAYER_JUMP_SPEED
        elif key == arcade.key.LEFT or key == arcade.key.A:
            self.player_sprite.change_x = -PLAYER_MOVEMENT_SPEED
        elif key == arcade.key.RIGHT or key == arcade.key.D:
            self.player_sprite.change_x = PLAYER_MOVEMENT_SPEED

    def on_key_release(self, key, modifiers):
        """Called when the user releases a key. """

        if key == arcade.key.LEFT or key == arcade.key.A:
            self.player_sprite.change_x = 0
        elif key == arcade.key.RIGHT or key == arcade.key.D:
            self.player_sprite.change_x = 0
```

Scrolling:

```
# How many pixels to keep as a minimum margin between the character
# and the edge of the screen.
LEFT_VIEWPORT_MARGIN = 250
RIGHT_VIEWPORT_MARGIN = 250
BOTTOM_VIEWPORT_MARGIN = 50
TOP_VIEWPORT_MARGIN = 100
```

```
        # Used to keep track of our scrolling
        self.view_bottom = 0
        self.view_left = 0
```

Coins and Sound addition

The codes in here adds coins that we can collect. When player hits any coins, or presses jump button it also adds sound.

```
# Coins
self.coin_list = arcade.tilemap.process_layer(my_map, coins_layer_name, TILE_SCALING)
```

```
# if you hit any coins
coin_hit_list = arcade.check_for_collision_with_list(self.player_sprite, self.coin_list)

for coin in coin_hit_list:
    if 'Points' not in coin.properties:   # özellikler
        print("Warning, collected a coin without a Points property.")
    else:
        points = int(coin.properties['Points'])
        self.score += points
    # Remove the coin
    coin.remove_from_sprite_lists()
    # Play a sound
    arcade.play_sound(self.collect_coin_sound)
```

Displaying the score.

This is the the “Score addition” part of my code. Now that we can collect coins and get points.It’s a very crucial and important features of a game structure. We all know that it’s impossible to describe game without a score or achievement.

```
    # For Showing Score
    score_text = f"Score: {self.score}"
    arcade.draw_text(score_text, 10 + self.view_left, 10 + self.view_bottom, arcade.csscolor.WHITE, 18)
```

Map Creation and Addition

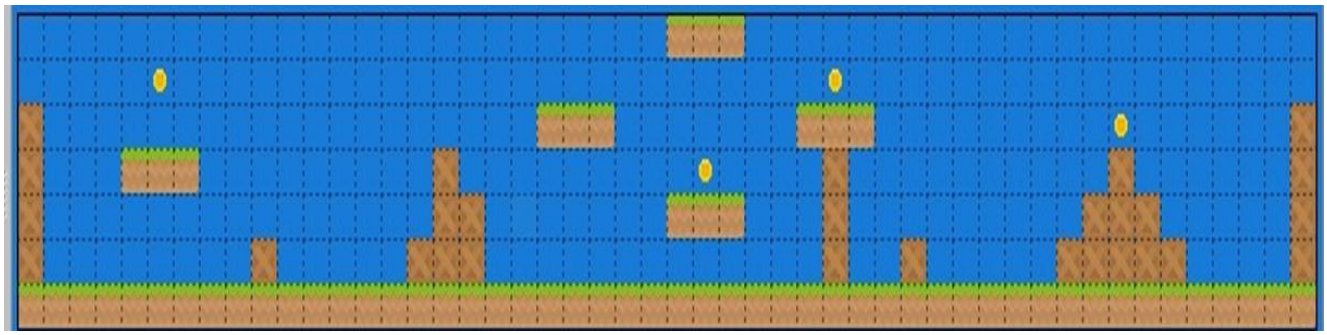
In order to determine the pathway of our game,we'll need map. Tiled is a 2D level editor that helps you develop the content of your game. Its primary feature is to edit tile maps of various forms, but it also supports free image placement as well as powerful ways to annotate your level with extra information used by the game. Tiled focuses on general flexibility while trying to stay intuitive. In here, we're adding the map that we've created into our code. We're coding tmx&tsx files here.

```
# The name of the map

level4new = ":resources:tmx_maps/map.tmx"
platforms_layer_name = 'Platformers'
coins_layer_name = 'Coins'
moving_platforms_layer_name = 'details'
dont_touch_layer_name = "Don't Touch"
my_map = arcade.tilemap.read_tmx(level4new)
self.end_of_map = my_map.map_size.width * GRID_PIXEL_SIZE
```

A Tiled map supports various sorts of content, and this content is organized into various different layers. The most common layers are the Tile Layer and the Object Layer. There is also an Image Layer for including simple foreground or background graphics. The order of the layers determines the rendering order of your content.

Layers can be hidden, made only partially visible and can be locked. Layers also have an offset, which can be used to position them independently of each other, for example to fake depth.

I want to show you our Level 1 map.It’s from resources of Arcade Library.



Character animation & definition

By writing this code, we’re defining our player’s character. And we made with an animation.

```
# Textures
main_path = ":resources:images/animated_characters/male_adventurer/maleAdventurer"

# Textures for idle standing
self.idle_texture_pair = load_texture_pair(f"{main_path}_idle.png")
self.jump_texture_pair = load_texture_pair(f"{main_path}_jump.png")
self.fall_texture_pair = load_texture_pair(f"{main_path}_fall.png")
# Textures for walking
self.walk_textures = []
for i in range(8):
    texture = load_texture_pair(f"{main_path}_walk{i}.png")
    self.walk_textures.append(texture)

# Initial texture (go to 0)
self.texture = self.idle_texture_pair[0]
self.set_hit_box(self.texture.hit_box_points)
```

We can see the image of our character.

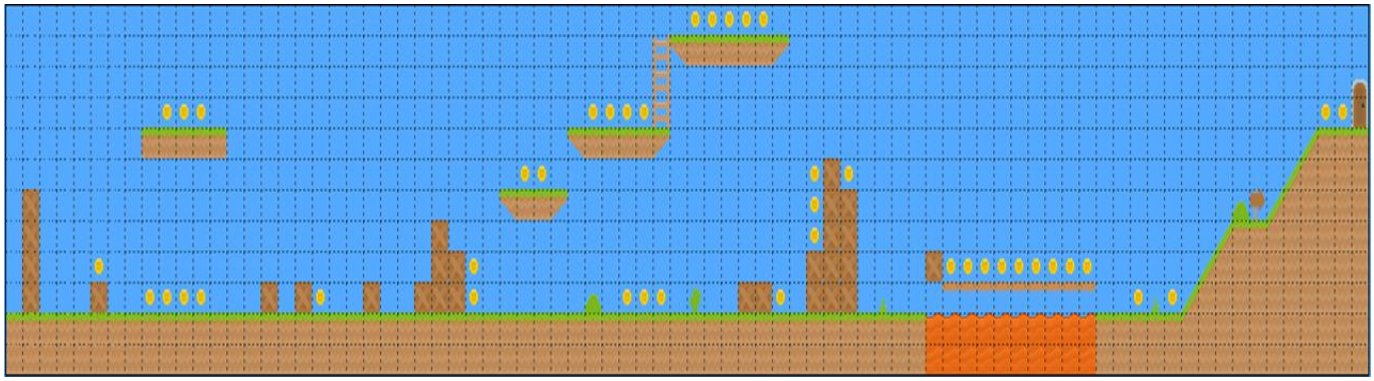
Prototype 2.

My game has the exact policy as the other casual platform games.In second prototype it hasn’t changed much.Normally the main character,a.k.a player has 3 health.But I've had some technical problems with my computer.That's why in this report,unfortunately I can't show you the "health" score or "healthbar". And also the game has some pits which you shouldn't fall.if you fall,you will die or you will probably lose one of your health.

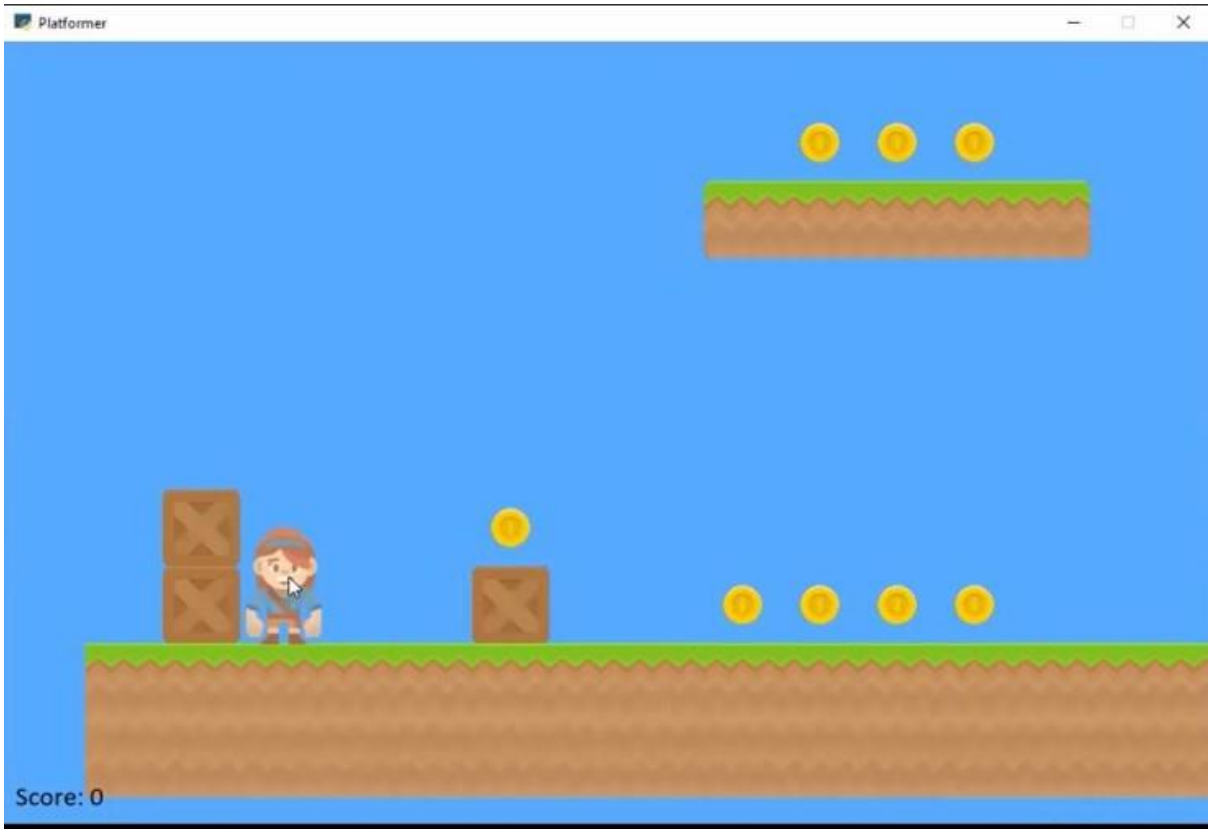
Loading our own map into game via “Tiled map editor”

This screenshoot is from "tiled"map editor. I developed more advanced version of Level 1.I've added different layers and objects, tiles in here. Player can connect to collect the coins, get a score,can climb a ladder, then can pass to the other level from door. These are the new tiles that I've added for new version. It contains "right sign table", "cactus", "bridge", "plant", "door" and "lava". My don't touch part is lava in here. If she fell into it or touch it, she'll die and lose one health. Some parts might be missing in game window. I'll handle it until final presentation. I couldn't make that code work properly because of my computer and bug errors. I think I'm still dealing with some issues about "map addition".

Level 1:

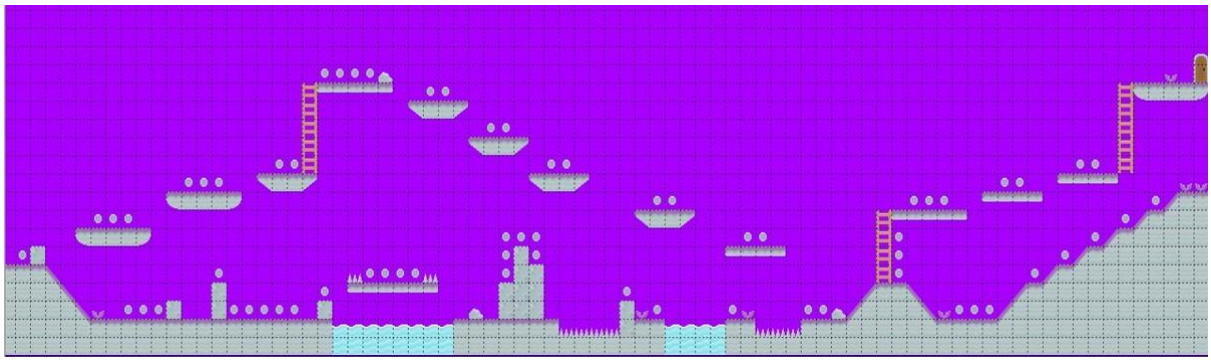


This is the runned version of Level 1.



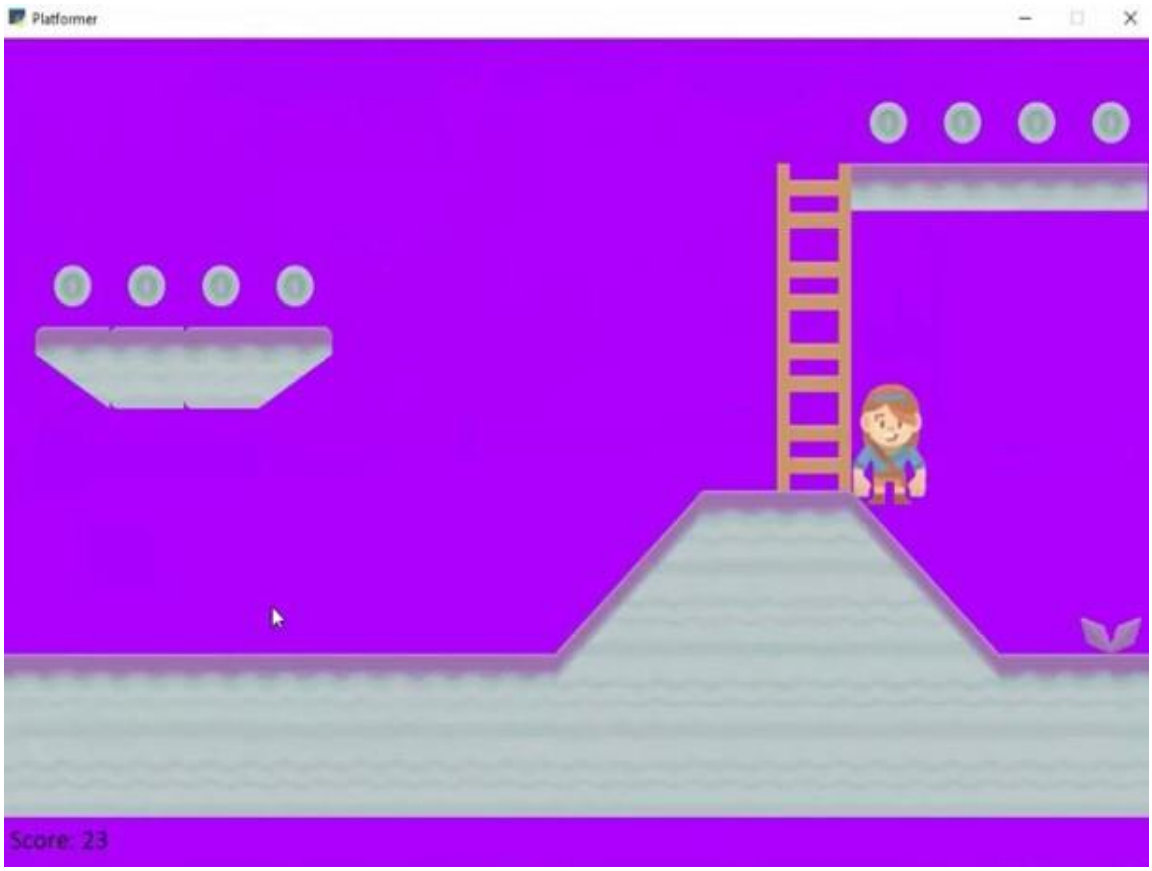
Level 2 description:

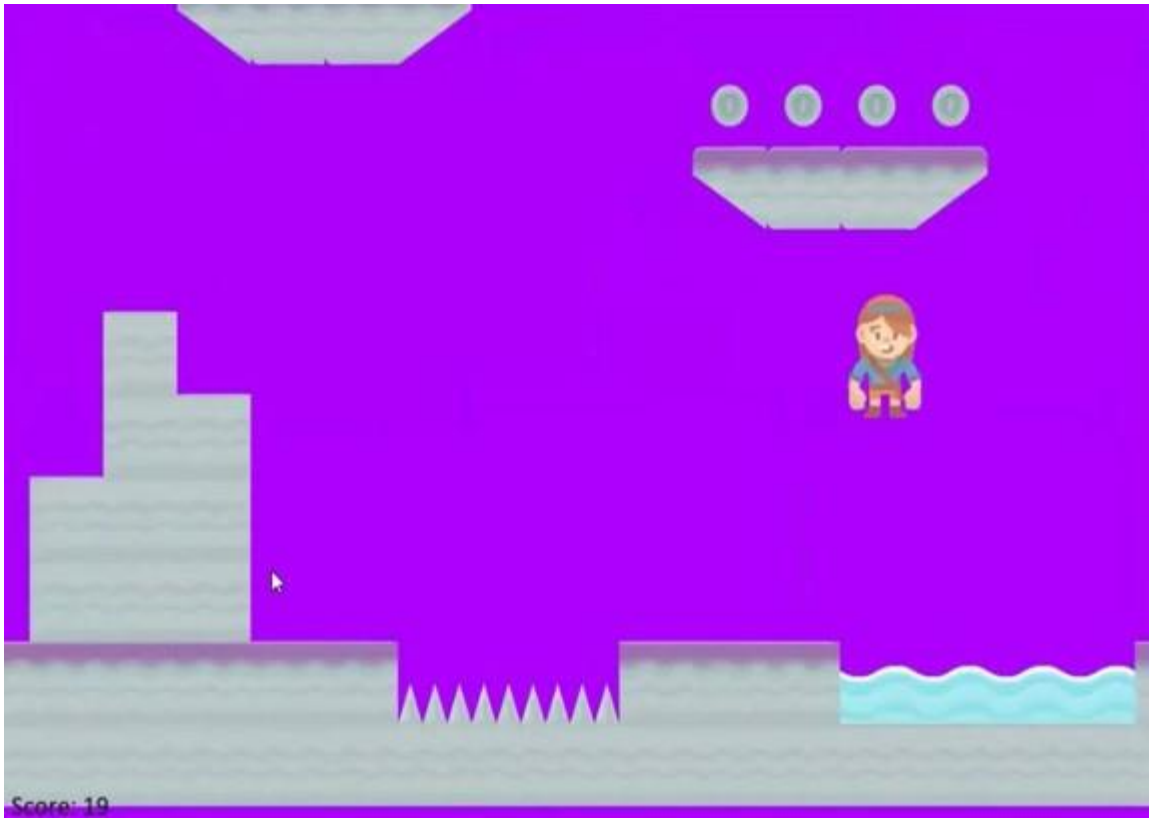
I’ve changed background colour and added ladders,spikes,rocks,grass,door,and water for “don’t touch” part.This map is more complex than the old one.Our character will collect coins and will get score.This game also has ladders,so character can climb with ladders and it will die if they fall on the spikes or water.



Level 2:

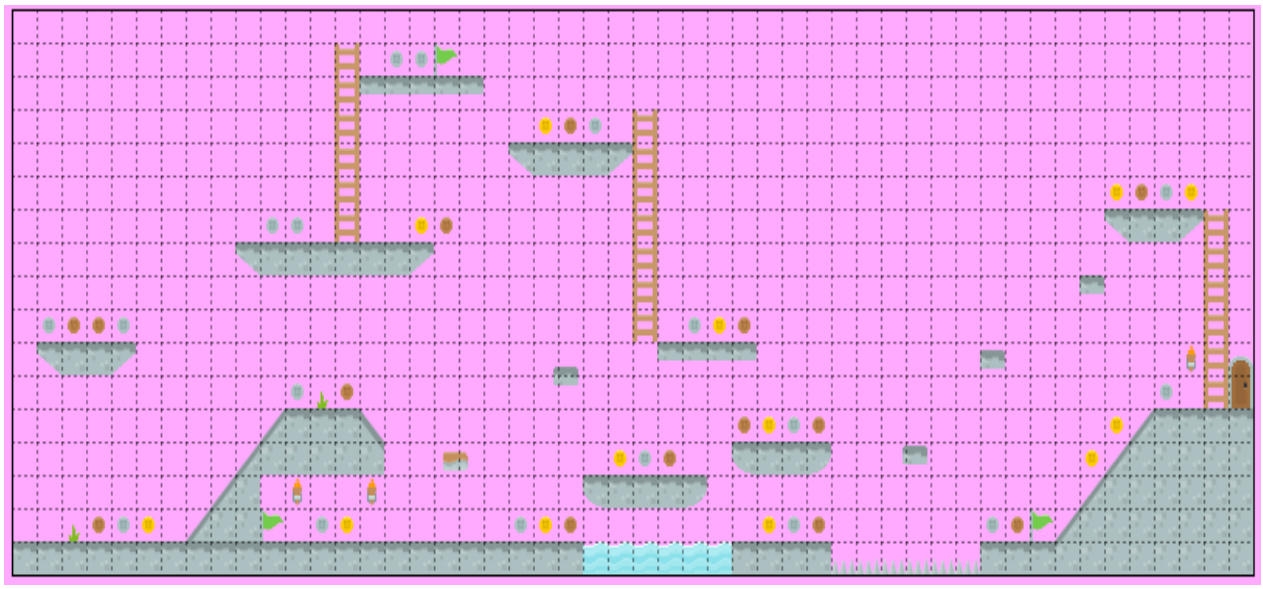
This is the runned version of Level 2.

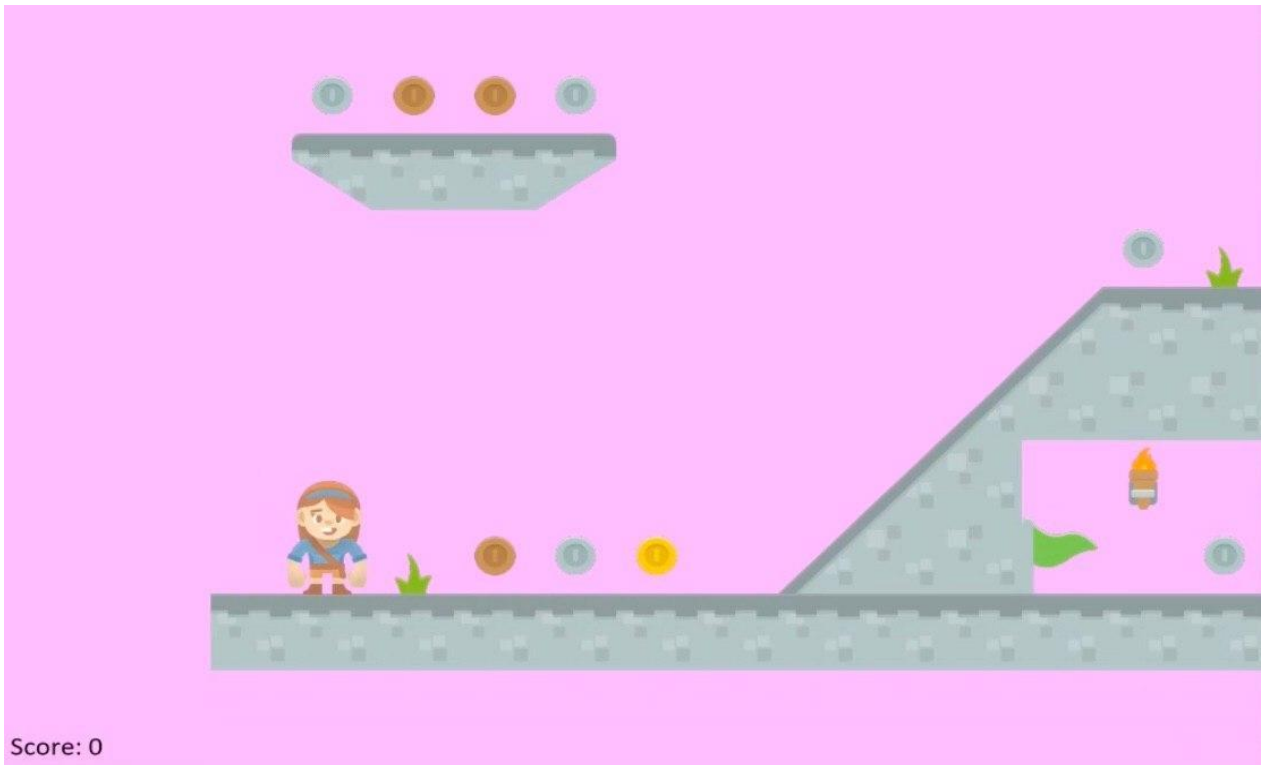




Level 3:

In Level 3 I achieved to add moving platforms.Also I added animated objects like flags.





Level and other layer addition

This adds foreground, background, and “Don’t Touch” layers.

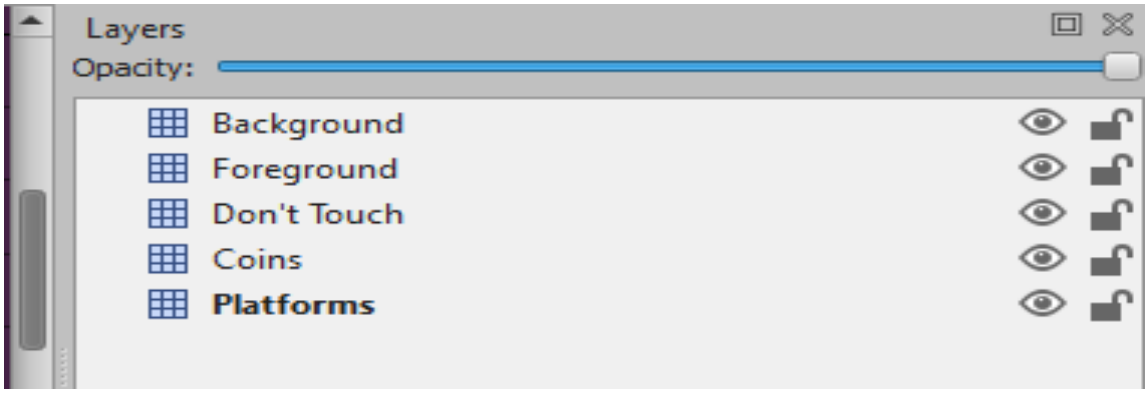
The background tiles appear behind the player.

The foreground appears in front of the player.

The Don’t Touch layer will reset the player to the start.

The player resets to the start if they fall off the map.

If the player gets to the right side of the map, the program attempts to load another layer.



Health addition

```
# Player Health
self.health = 3
```

```
    # Set the camera
    self.view_left = 0
    self.view_bottom = 0
    changed_viewport = True
    arcade.play_sound(self.game_over)
    self.health -= 1

if self.health == 0:
    arcade.pause(0.3)
    self.view_left = 0
    self.view_bottom = 0
    changed_viewport = True
    self.health = 3
```

Level:

In order to add level we have to code our map just like a resource map:

```
        # Map name
        map_name = f":resources:tmx_maps/map2_level_{level}.tmx"
```

```
        # Level
        self.level = 1

        # Load sounds
        self.collect_coin_sound = arcade.load_sound(":resources:sounds/coin1.wav")
        self.jump_sound = arcade.load_sound(":resources:sounds/jump1.wav")
        self.game_over = arcade.load_sound(":resources:sounds/gameover1.wav")

    def setup(self, level):
```

```
        # See if the user got to the end of the level
        if self.player_sprite.center_x >= self.end_of_map:
            # Advance to the next level
            self.level += 1

            # Load the next level
            self.setup(self.level)

            # Set the camera to the start
            self.view_left = 0
            self.view_bottom = 0
            changed_viewport = True
```

Final Product Description

& Problems that I’ve faced during project

Unfortunately the final product that I’ve made is not the exact thing in my mind. Animations,health,don't touch parts.Like I said it's because of technical and debugger problems.I've tried to fix that.For now this I just only came this far.Of course this is not the exact thing was on my brain about this project.But it's better than nothing.I'll believe that in future,I'll able to create more professional and advanced games.But personally I enjoyed from this project so much.I’ve learned a lot of things from this course.I’ll definetely will make a game in the future.Once again I realized being an engineer is my thing.I recommend everyone to learn and work with Python,especially who want to work in game development.

Also I want to talk about the problems that I’ve faced while working in game developing.I think it’s because of my computer,but also I had some bugs and errors about my codes.Some sample codes that I copied from internet didn’t worked in my PC at all.It was the codes which written by creator of Arcade library.

Resources,references training video addresses.

Mostly we used “arcade.academy”. It was pretty useful. And it was our teacher’s recommendation.

From Youtube we watched a plenty of videos from channel called “Tech with Tim”, ”DaFluffyPotato”, “Professor Craven” and etc.