ALGORITHM AND PROGRAMMING FINAL PROJECT

PROJECT DOCUMENTATION

WRITTEN BY:

NAME : FELISE AMORE PANDIORA
CLASS : L1BC
MAJOR : COMPUTER SCIENCE

BINUS INTERNATIONAL UNIVERSITY
CENTRAL JAKARTA
2023

Table Of Contents

TABLE OF CONTENTS…….

DESCRIPTION……..1

Introduction……1

Purpose of program……1

DESIGN…….2

Use-case Diagram……..2

Activity Diagram……3

Class Diagram……6

IMPLEMENTATION……..…8

Functions used in program……...……...…8

Modules used in program……...……...…11

REFLECTION……...……...…13

EVIDENCE OF A WORKING PROGRAM……...…14

DESCRIPTION

1. Introduction

This program is a simple jackpot slot machine program built using pygame python. The player will first be brought to the starting page where the player will see two buttons, start and quit respectively. The button functions as what their name implies. To quit, click on the quit button; and to start, click on the start button.

Arriving at the game page, the player will see a 2D slot machine with quite a lot of buttons, from left to right, there are 10, 20, reset, spin, quit, 50, and 100. The player will be given the starting amount of cash and jackpot. To play, simply click on the 'spin' button; To reset the cash and jackpot, click on the reset button; To quit the game, click on the quit button. Players can also choose how much cash to bet on, just click on one of the bet buttons ($10 (default), $20, $50, $100). The game will go on until the player runs out of cash.

The scoring system in this slot machine is:

ICONS	DOUBLE ALIGN	TRIPLE ALIGN
Sad Face	x0	x0
Bell	x1	x10
Cherry	x2	x20
Melon	x2	x30
Orange	x2	x100
Grape	x2	x200
Diamond	x5	x300
Seven	x10	x1000

Notes: The bet money will come back to the player if and only if no sad face appears among the three icons.

2. Purpose of Program

This program is made to complete the Algorithm and Programming final project. For this final project, students are expected to make an application using the knowledge and skills we have learned in Python over the semester. One of the must include topics is class. Of course students are not limited to only use the topics taught in the syllabus, on the other hand, we are encouraged to use topics outside of it.

As I have learnt Python before entering Binus, one of the topics that is new and interesting to me is Pygame, which is the reason why I wanted to make an application using Pygame, considering it as a challenge for myself. After ruminating numerous ideas, I decided to make a Jackpot Slot machine using Pygame.

DESIGN

1. Use-case Diagram

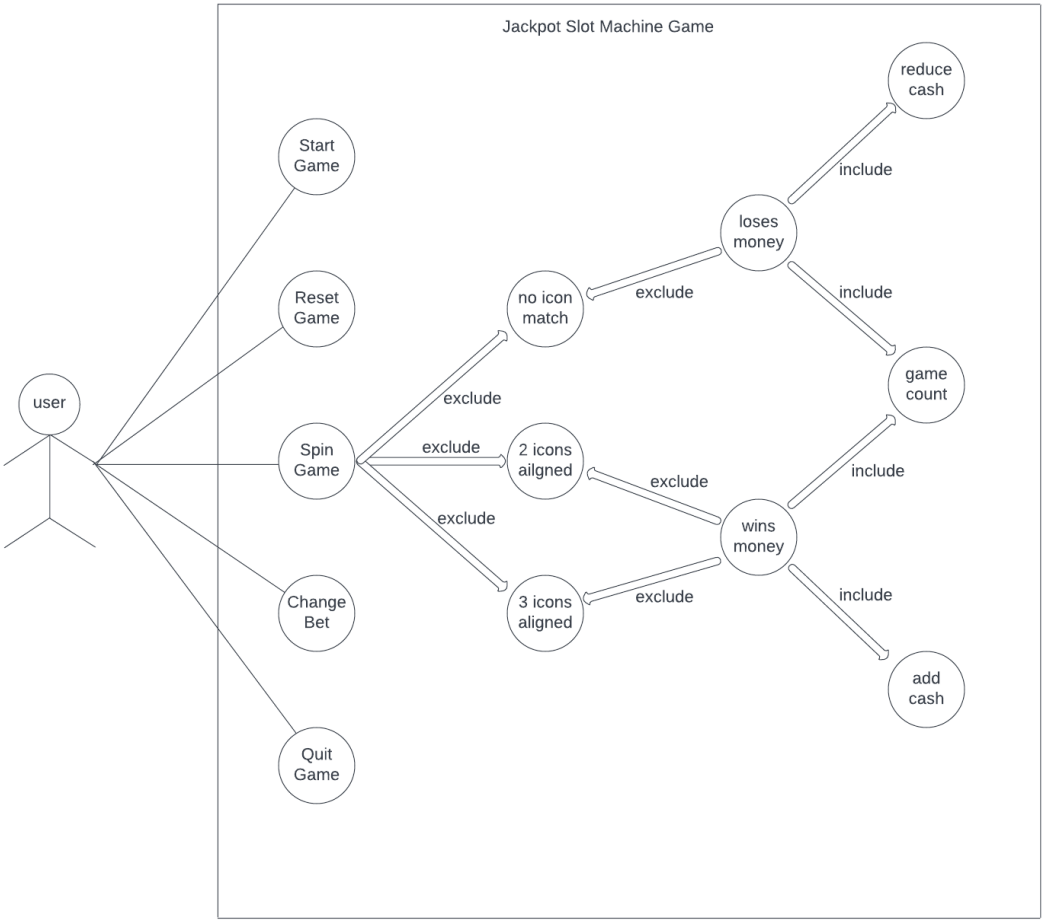

fig 1.1 Use-case Diagram for SlotMachineGame

A little explanation for the use-case diagram above, so some of the features that the player can do is to start the game, reset the game, spin the game, change the bet amount, and quit the game.

By doing the spin game, the player will either get no matching icon, two matching icons, or all matching icon from the shuffle process. If none of the icon matches, the player will not win any prize, then the game will automatically reduce the player's cash by the amount the player lost and increase the game count by 1. However, if among those icons appear the icon seven, the player will win jackpot (prize) as it is the wildcard set by the program.

If the player matches two or all icons, they will win prize according to the value of the icon they match, then the game will automatically add the winning prize into their cash and increase the game count by 1.

As for the bet money the user paid to spin the game, it will return to the player if no sad face icon appears on the reel.

2. Activity Diagram
   - Start Page

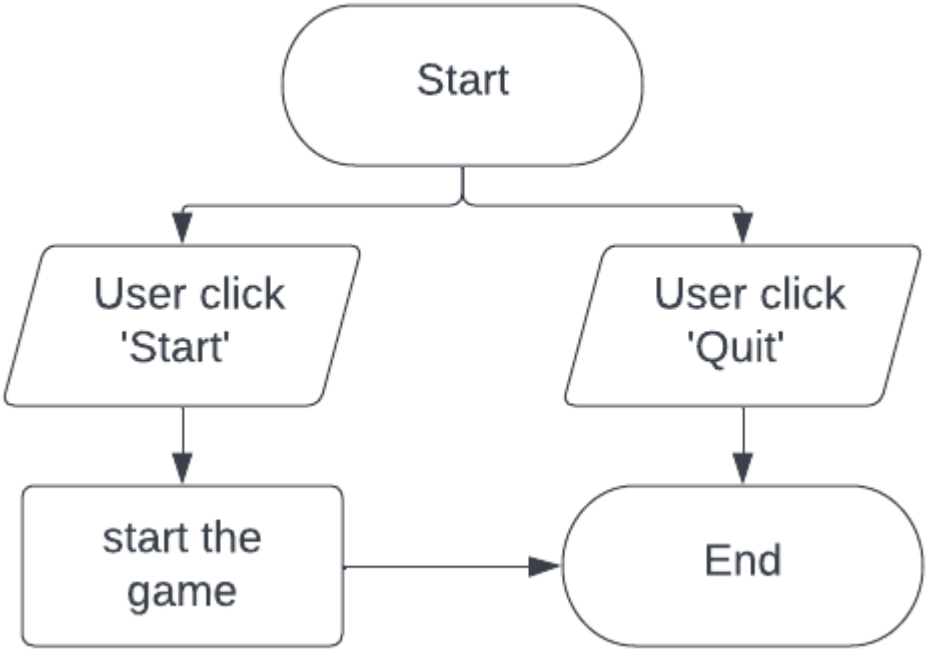

Fig 2.1 Activity Diagram for the starting page

   - Game Page (Reset Button)

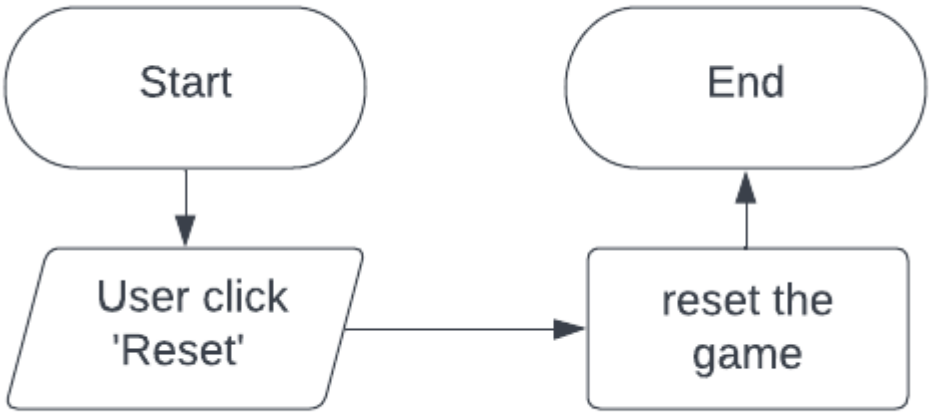

fig 2.2 Activity Diagram for Reset Button in Game Page

- Game Page (Spin Button)

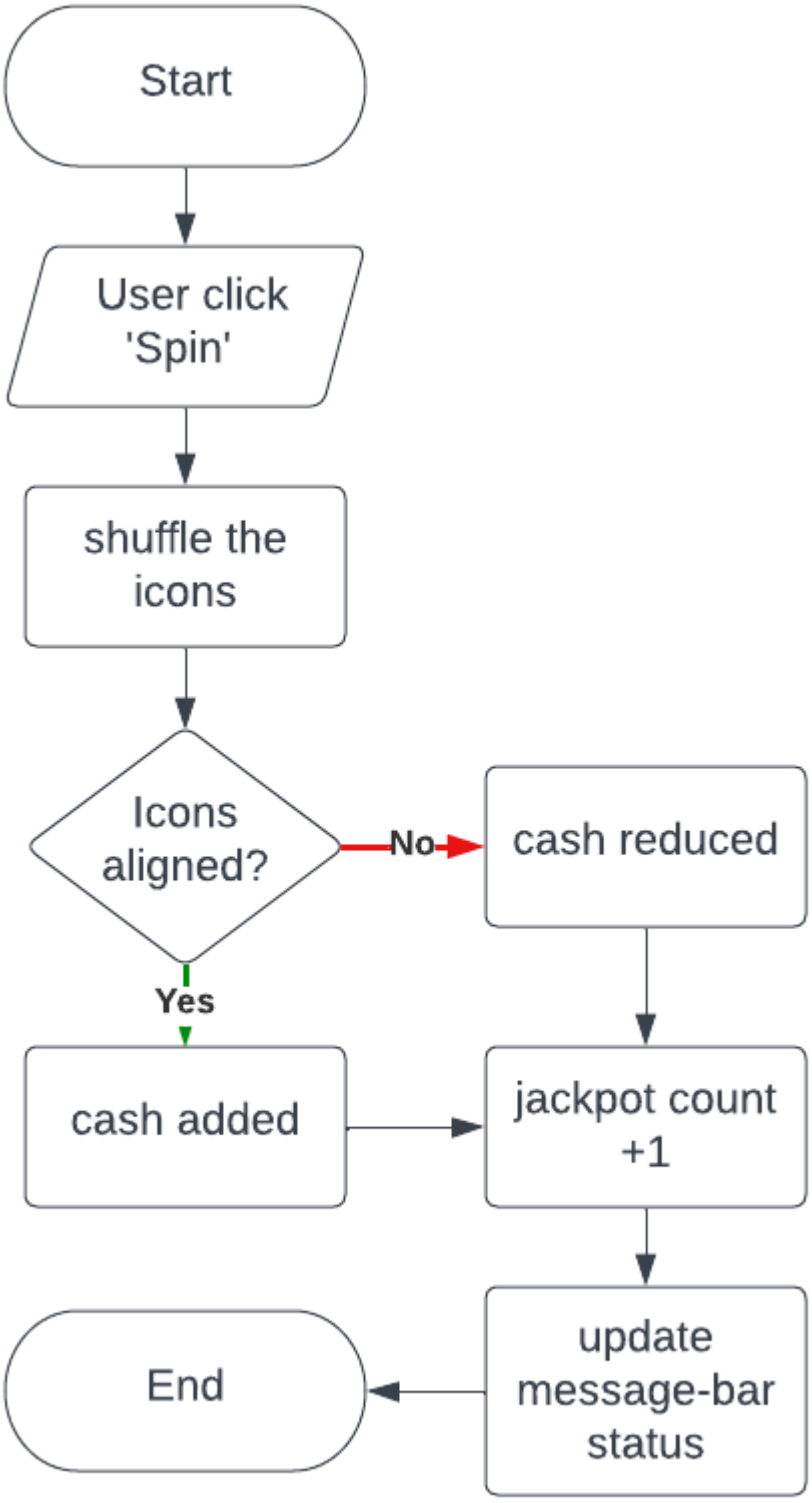

fig 2.3 Activity Diagram for Spin Button in Game Page

- Game Page (Bet Button)

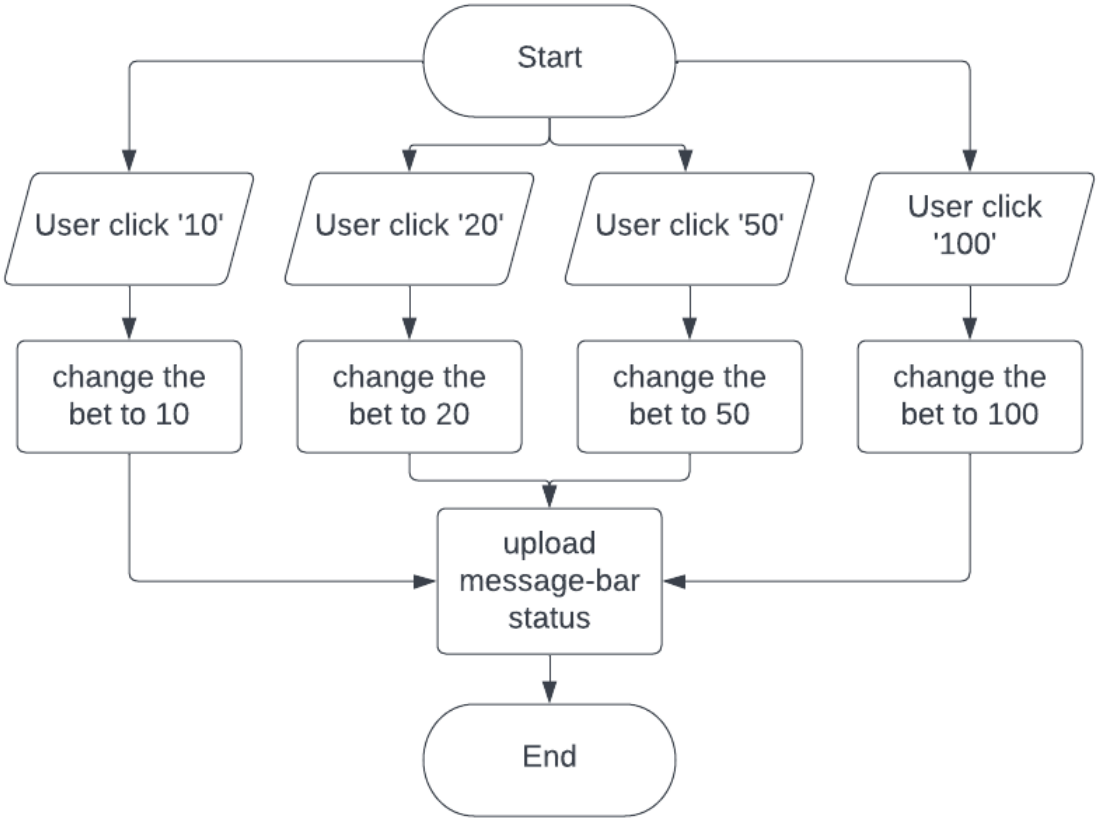

fig 2.4 Activity Diagram for Bet Buttons in Game Page

- Game Page (Quit Button)

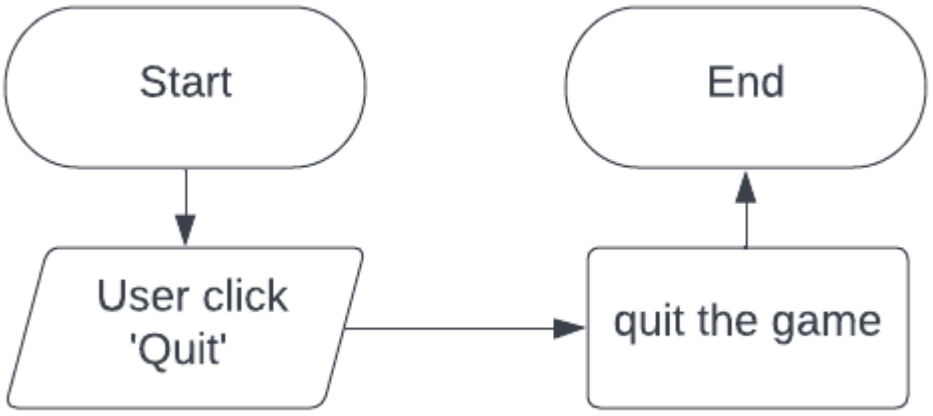

fig 2.5 Activity Diagram for Quit Button in Game Page

3. Class Diagram

SlotMachine
- starting_jackpot - starting_cash
+ set_resettable_values() + __create_icons() + set_bet(bet) + get_bet() + get_current_cash() + get_current_jackpot() + get_current_message() + spin() + __pay() + __increase_jackpot() + __check_results() + jackpot_win() + reset()

fig 3.1 Class Diagram for SlotMachine

SlotMachineButton
- image_file_name - pos

fig 3.2 Class Diagram for SlotMachineButton

SlotMachineBetButton (derived from SlotMachineButton)
- image_file_name - bet_value - pos

fig 3.3 Class Diagram for SlotMachineBetButton

SlotMachineActionButton (derived from SlotMachineButton)
- image_file_name - function_callable - pos
+ execute_func()

fig 3.4 Class Diagram for SlotMachineActionButton

DigitalFont
- method - pos - color = (196, 65, 46)
+ get_rendered_surface() + update()

fig 3.5 Class Diagram for DigitalFont

Icon
- name - win_rate_full - win_rate_two - icon_image - bonus_win_rate = 0

fig 3.6 Class Diagram for Icon

Start
+ start()

fig 3.7 Class Diagram for Start

IMPLEMENTATION

1. Functions used in Program

btn.py -> This file contains functions to set up the buttons for the game, which are used in *start.py*. This file used the *pygame* module.

- __init__(self, mage_file_name, pos):
  - Load the png image with transparency
  - Move the rect to make sure it follows the image
- __init__(self, mage_file_name, bet_value, pos):
  - Set the bet value
- __init__(self, mage_file_name, function_callable, pos):
  - Set functioning button
- execute_func(self):
  - Execute the function callable

font.py -> This file contains functions to set up everything related to text and/or font in the game, which are used in *start.py*. This file used the *pygame* module.

- __init__(self, method, pos, color = (196, 65, 46)):
  - Set the font type
  - Set the font color
  - Set the method
  - Set the position
- get_rendered_surface(self):
  - Returns the rendered text
- update(self):
  - Render the text by update

icon.py -> This file contains a function to set up the icons shown on the reels, which is used in *slotmachine.py*. This file used the *pygame* module.

- __init__(self, name, win_rate_full, win_rate_two, icon_image, bonus_win_rate = 0):
  - Set icons to be shown in the reels
  - Set win rate when icons are fully aligned
  - Set win rate when only 2 icons are aligned
  - Set the bonus win rate of icons

slotmachine.py -> This file contains functions to set up everything required when playing the slot machine, which are used in *start.py*. This file used functions from *icon.py* and modules like *pygame* and *random*.

- __init__(self, starting_jackpot, starting_cash):
  - Set all sounds that will be used in the game
  - Set the starting cash and jackpot values
  - Call function to create icons
  - Call the method used to set the initial values
- set_resettable_values(self):
  - Set the values of slot machine which are resettable
- __create_icons(self):
  - Set the icons in array
- set_bet(self, bet):
  - Condition if cash is valid, users are allow to bet
  - Condition if cash is lower than 0, it will tell user that slot machine is out of cash
- get_bet(self):
  - Only to get attributes
- get_current_cash(self):
  - Only to get attributes
- get_current_jackpot(self):
  - Only to get attributes
- get_current_mesages(self):
  - Only to get attributes
- spin(self):
  - Check if cash is enough to spin
  - Call function to reduce money whenever user spin
  - Call function to increase jackpot by 1 whenever user spin
  - Set the chance of icons appearing
  - Call function to check result
  - Show message when slot machine is unable to spin
- __pay(self);
  - Reduce cash by bet amount every time user spin
- __increase_jackpot(self):
  - Increase the jackpot prize to be winned
- __check_results(self):
  - Check how much player has won or lost
  - Check how many certain icon is shown on reel then multiply win rate to bet and add it to winnings
  - Play jackpot when three icons aligned but not sad face
  - If there is one seven icon, it is considered a win
  - If there is no sad face icon, it is considered bet return
  - Set appropriate messages whenever user win or lost or error

- jackpot_win(self):
  - Return the value of user's jackpot winnings
  - Set the wildcard jackpot number and generate number from 1-100
  - Compare the wildcard to the number. If match, then user wins then set the current jackpot as winnings and reset the jackpot
- reset(self):
  - Reset the slot machine when started

start.py -> This file contains a function to start the game, which is used in *main.py*. This file used functions from *slotmachine.py*, *font.py*, and *btn.py* and modules like *pygame*, *time*, and *random*.

- start_game():
  - Assigning the display variables
  - Create the slot machine object and hashes to be used by the game
  - Set the current icon images or spin result icons
  - Create text labels
  - Create bet buttons
  - Create action buttons
  - Create all icons to be shown in reel
  - Set the game clock
  - Save the reels position in an array with tuples
  - Add the spin result symbols to icon images
  - Set the variables to be used by the game loop
  - Set the previous spin result
  - Set the current slot machine values as previous values
  - Create functions to get the precious slot machine values
  - Set text values
  - Creating a loop for the game

main.py -> This file is where the game can be run. This file used functions from *start.py* and the *pygame* module.

- __init__(self):
  - Set the screen settings
- start(self):
  - Set the start and quit button
  - Create a loop for the starting page

2. Modules used in Program

- Pygame

  Pygame is the module used to make this slot machine game. As I am not really familiar with the Pygame module yet, I had to read and watch quite a lot of explanations of how to implement it into the game. Turns out, using Pygame made it a lot easier for me to create this app. It helps me in setting the background music, setting the clock, setting the buttons, etc. Here are some snippets of codes using pygame module:

```
# To set all sounds
pygame.mixer.init()
self.bet_snd = pygame.mixer.Sound("sounds/bet_snd.wav")
self.bet_no_cash_snd = pygame.mixer.Sound("sounds/bet_no_cash_snd.wav")
self.spin_snd = pygame.mixer.Sound("sounds/spin_snd.ogg")
self.spinning_snd = pygame.mixer.Sound("sounds/spinning_snd.ogg")
self.reset_snd = pygame.mixer.Sound("sounds/reset_snd.ogg")
```

fig 4.1 Snippet of pygame module

```
# Set the game clock
clock = pygame.time.Clock()
```

fig 4.2 Snippet of pygame module

```
# Setting the start button
start = pygame.image.load('images/start.png')
start = pygame.transform.scale(start, (200, 110))
# Setting the quit button
exit = pygame.image.load('images/exit.png')
exit = pygame.transform.scale(exit, (200, 110))
```

fig 4.3 Snippet of pygame module

- Random

  Random module is the module I use to shuffle the icons, display and get random icons. Here is the snippet of codes using random module:

```
# To display a random icon in each reel and play the spinning sound
for i in range(3):
  screen.blit(icons[random.randint(0, len(icons) - 1)].image, reel_positions[i])
slot_machine.spinning_snd.play()
```

fig 4.4 Snippet of random module

```
# Generate a random number from 1 to 100
jackpot_try = random.randint(1, 100)
```

fig 4.5 Snippet of random module

- Time

  Time is the module I use to represent and get time in my codes. Here is the snippet for that:

```
# Set the start time to current time. Making the spin animation run
start_time = time.time()
```

fig 4.6 Snippet of time module

REFLECTION

During the process of making this jackpot slot machine application, I met a lot of difficulties which I needed external help to solve. Firstly, when I have no idea how to make a certain thing, I would either find the solution on google or youtube videos. This is actually not hard as nowadays the internet has almost everything covered, however, sometimes a lot of time is needed to find the solution. Secondly, along the way of coding, not a few errors occurred. Usually for codes, I can find the solution in stackoverflow website.

What I learnt through making this application is to not panic easily as some problems actually have really simple solutions but are made hard by our own overthinking. And I found that I am the person who needs some time to finish making something because my inspiration cannot come in a short amount of time.

Regarding the modules, I am now able to understand how to implement them into codes more than before as I was also learning while making this application. However, through this process, I also realised that I still need to learn more about pygame as it has too many functions for me to learn in this short amount of time.

In my opinion, this slot machine game can still be improved in some way. For now, some of the improvements I can think of are; a feature to make the user able to set their own starting cash, and animation of pulling down slot triggers instead of clicking buttons, etc.

EVIDENCE OF A WORKING PROGRAM

Start Page

fig 5.1 Game Starting page

Game Page

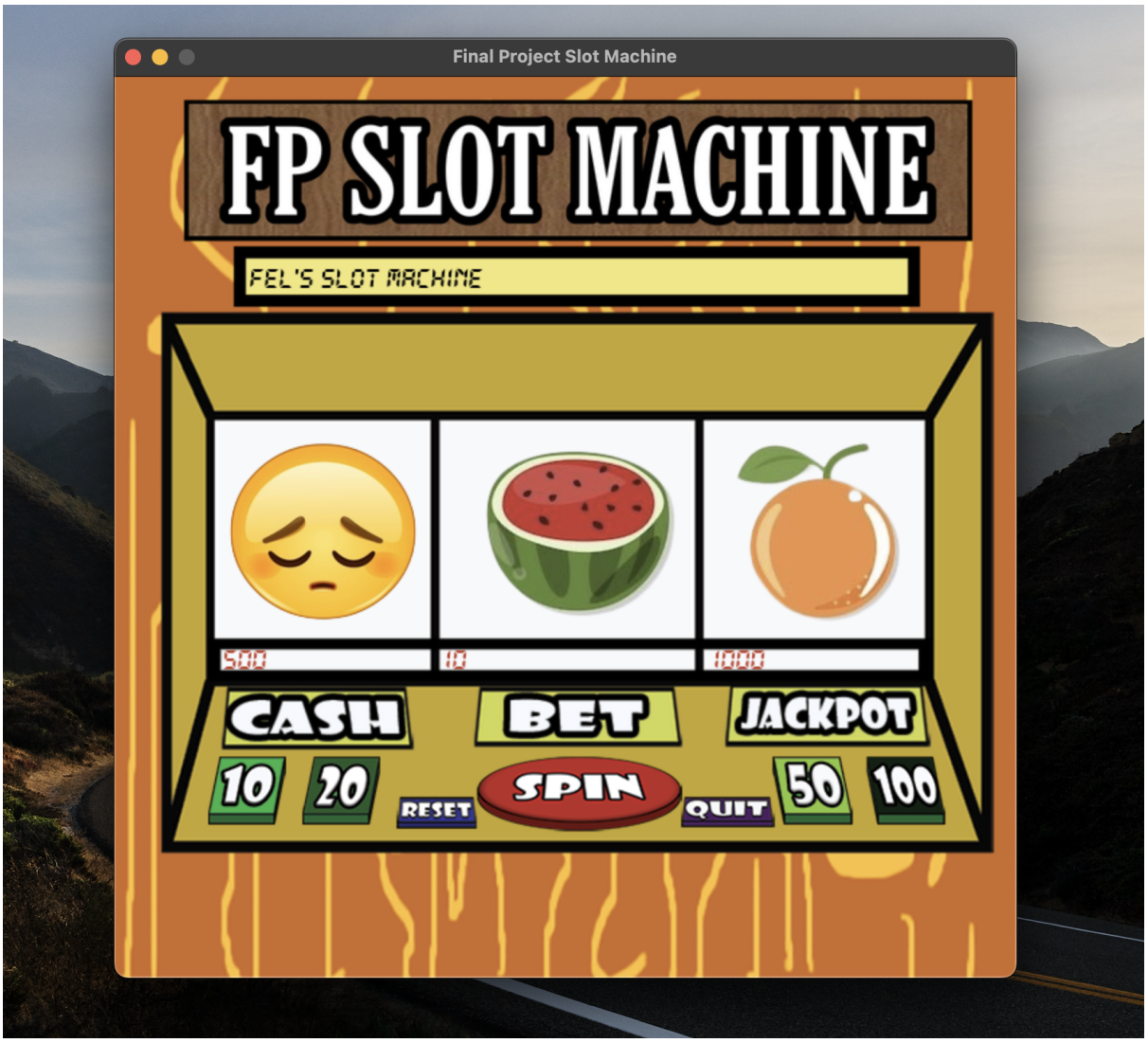

fig 5.2 Game Main Page

Losing Money scenario

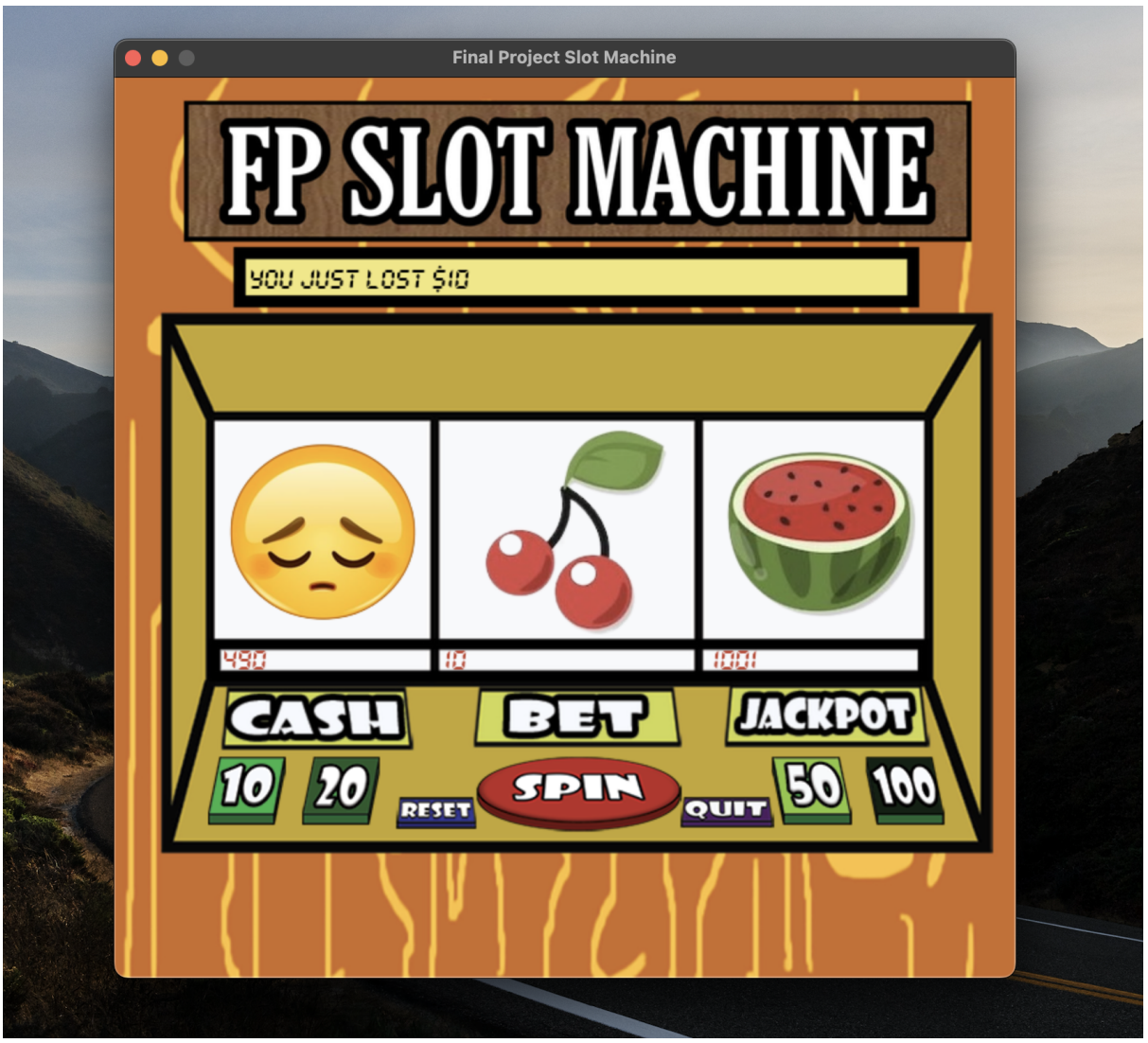

fig 5.3 Scenario where player lose money

Changing Bet Amount scenario

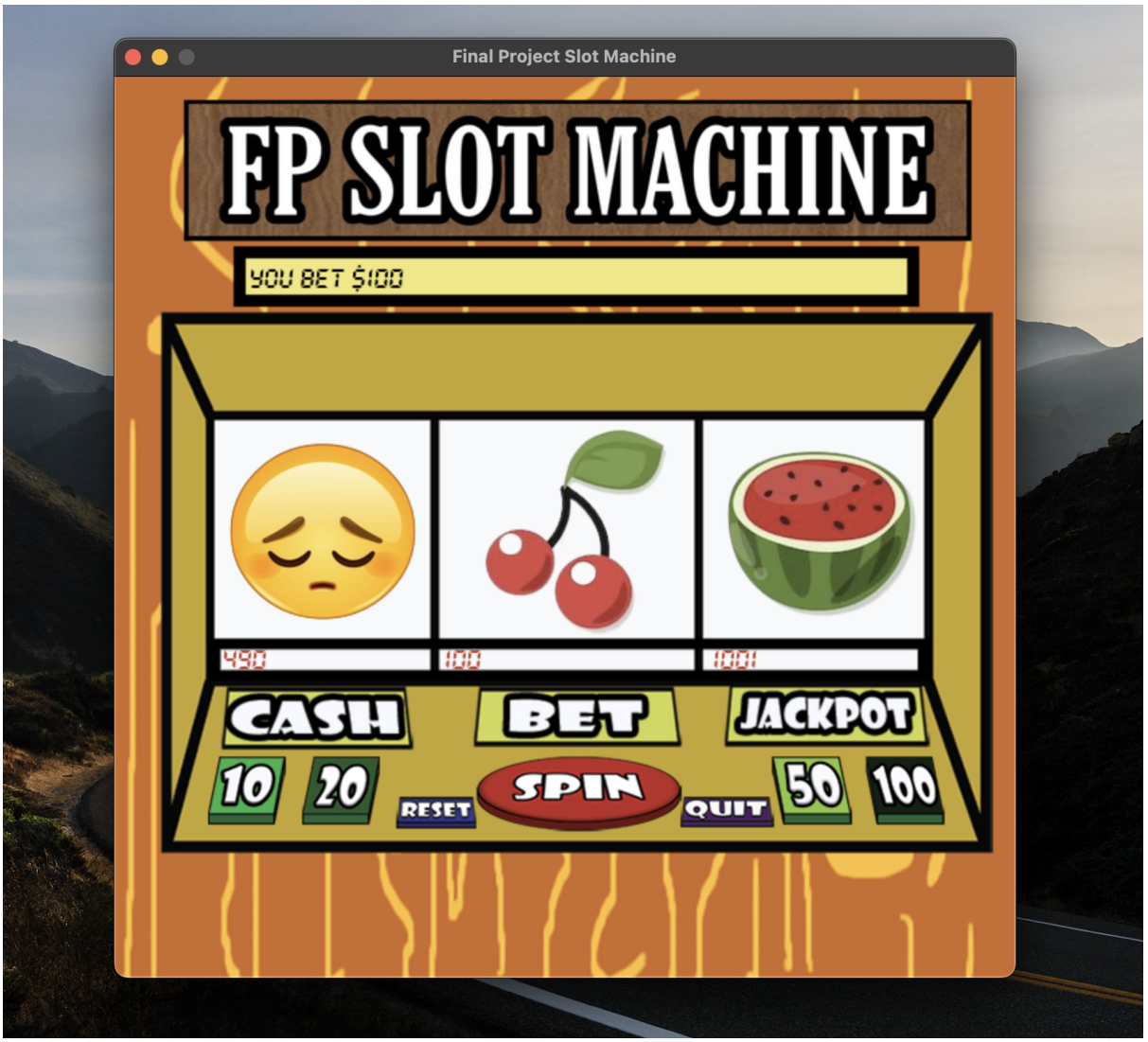

fig 5.4 Scenario where player change bet amount

Winning Money scenario

fig 5.5 Scenario where player win money