CATCH THE COIN

Game Design Development

King Awhaetoma

Overview

"Catch the Coin" will be a basic 2D arcade game to demonstrate the overall flow of a game using pygame and simpleGE.

The premise is extremely simple: The player is a boy named Kross, the boy character Kross appears near the bottom of the gameplay screen with a background image of a Beach side him. The user can move Kross to the left and right with the corresponding arrow keys.

A series of coins fall from the top of the screen. Each coin will fall from a different x position, and at a different speed between 3 and 8 pixels per frame straight down.

If Kross touches a coin, a positive sound effect is played. If a coin leaves the bottom of the screen, it is reset to a new random position at the top of the screen and a new falling speed. The game continues for a set period (ten seconds for playtesting purposes)

Features:

1. Player Control: The player can move the character left or right using the arrow keys on the keyboard.
2. Coin Collection: Coins fall from the top of the screen, and the player collects them by colliding with them using the character sprite.
3. Sound Effects: Sound effects are played when a coin is collected, enhancing the gameplay experience.
4. Background Music: Background music adds ambiance to the game and keeps the player engaged.

Gameplay

Controls

- Left Arrow Key: Move Kross to the left.
- Right Arrow Key: Move Kross to the right.

Objectives

- **Collect as many coins as possible within the given time limit.**
- **Avoid colliding with obstacles or boundaries.**

Scoring

- **Each collected coin adds to the player's score.**
- **The final score is based on the number of coins collected.**

Game Over

- **The game ends when the timer runs out.**
- **The final score is displayed, and the player can choose to play again or quit.**

Assets

- Characters:
    - Kross: The player-controlled character.
- Items:
    - Coin: Collectible items scattered across the game area.
- Backgrounds:
    - Plain field background for the game scenes.
- Sounds:
    - Coin collection sound effect.
    - Background music.
- User Interface:
    - Score label.
    - Timer label.
    - Instructional text.

Scenes

Instruction Scene

- Displays game instructions and controls.
- Provides options to start the game or quit**.**

Game Scene

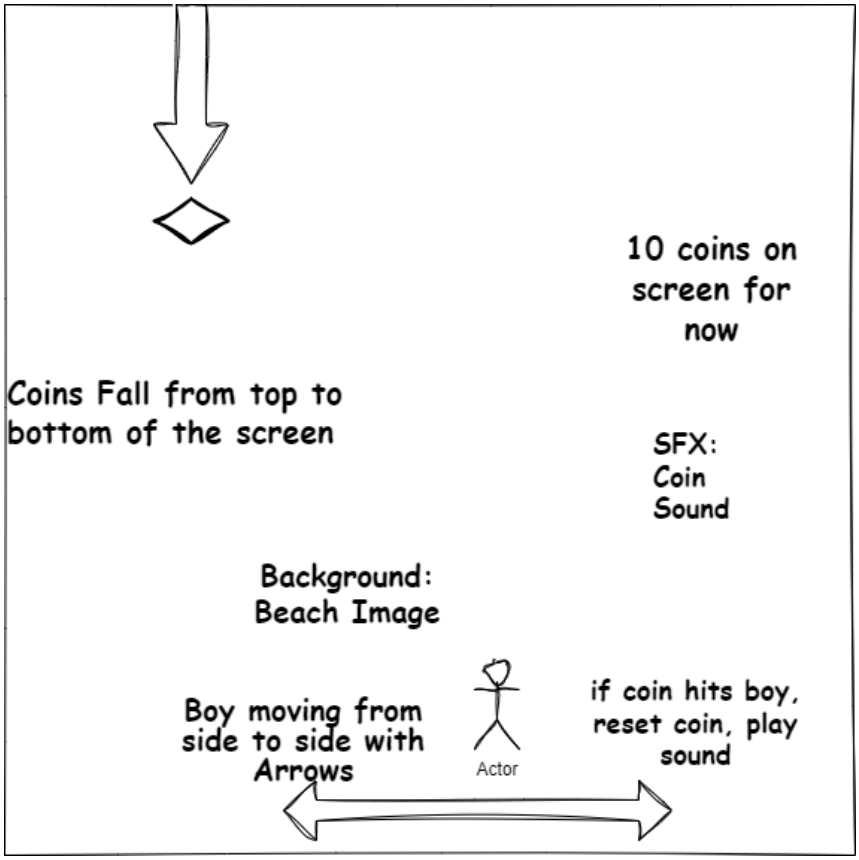
- Main gameplay scene where Kross collects coins.
- Displays Kross, coins, score, and timer.

Code Structure:

The code is organized into three main classes:

1. Coin Class: Represents the coins that fall from the top of the screen. It inherits from the simpleGE.Sprite class and includes methods to reset the coin's position, check for collisions, and play a sound effect when collected.
2. Boy Class: Represents the player-controlled character. It also inherits from the simpleGE.Sprite class and includes a method to handle user input for moving the character left or right.
3. Game Class: Represents the game scene. It inherits from the simpleGE.Scene class and includes methods to initialize the game, process game logic, and handle collisions between the character and coins.

Game Play Background




```
# Import necessary libraries and modules
import pygame
import random
import simpleGE


# Define the Coin class
class Coin:
    # Constructor to initialize the coin
    def __init__(self, scene):
        # Initialize the coin properties
        self.setImage("Coin.png")
        self.setSize(25, 25)
```

```
        self.reset()
        self.coinSound = simpleGE.Sound("coinEffect.wav")

    # Method to reset the position of the coin
    def reset(self):
        self.y = 10
        self.x = random.randint(0, self.screenWidth)
        self.dy = random.randint(3, 8)

    # Method to check if the coin is out of bounds
    def checkBounds(self):
        if self.bottom > self.screenHeight:
            self.reset()

# Define the Boy class
class Boy:
    # Constructor to initialize the boy character
    def __init__(self, scene):
        # Initialize the boy properties
        self.setImage("boy.png")
        self.setSize(90, 90)
        self.position = (320, 400)
        self.moveSpeed = 5

    # Method to process the movement of the boy character
    def process(self):
        if self.isKeyPressed(pygame.K_LEFT):
```

```
            self.x -= self.moveSpeed
        if self.isKeyPressed(pygame.K_RIGHT):
            self.x += self.moveSpeed

# Define the LblScore class
class LblScore:
    # Constructor to initialize the score label
    def __init__(self):
        # Initialize the score label properties
        self.text = "Score: 0"
        self.center = (120, 30)

# Define the LblTime class
class LblTime:
    # Constructor to initialize the time label
    def __init__(self):
        # Initialize the time label properties
        self.text = "Time Left: 10"
        self.center = (500, 30)

# Define the Game class
class Game:
    # Constructor to initialize the game scene
    def __init__(self):
        # Initialize the game scene properties
        self.setImage("plainField.png")
        self.timer = simpleGE.Timer()
```

```
        self.timer.totalTime = 10
        self.score = 0
        self.boy = Boy(self)
        self.coins = [Coin(self) for _ in range(10)]
        self.lblScore = LblScore()
        self.lblTime = LblTime()
        self.sprites = [self.boy, self.coins, self.lblScore, self.lblTime]
        pygame.mixer.music.load("newSong.mp3")
        pygame.mixer.music.set_volume(0.3)
        pygame.mixer.music.play(-1)

    # Method to process the game logic
    def process(self):
        for coin in self.coins:
            if coin.collidesWith(self.boy):
                coin.coinSound.play()
                coin.reset()
                self.score += 1
                self.lblScore.text = f"Score: {self.score}"
        self.lblTime.text = f"Time Left: {self.timer.getTimeLeft():.2f}"
        if self.timer.getTimeLeft() < 0:
            print(f"Final Score: {self.score}")
            self.stop()

# Define the Instruction class
class Instruction:
    # Constructor to initialize the instruction scene
```

```
def __init__(self, score):
    # Initialize the instruction scene properties
    self.setImage("plainField.png")
    self.response = "quit"
    self.instruction = simpleGE.MultiLabel()
    self.instruction.textlines = [
        "Welcome to Kross's Coin Collector!",
        "You are playing as Kross, the adventurous boy.",
        "Use the left and right arrow keys to move Kross.",
        "Your objective is to collect as many coins as possible in 10 seconds.",
        "",
        "Good luck and have fun!"]
    self.instruction.center = (320, 240)
    self.instruction.size = (500, 250)
    self.prevScore = score
    self.lblScore = simpleGE.Label()
    self.lblScore.text = f"Last Score: {self.prevScore}"
    self.lblScore.center = (320, 50)
    self.btnPlay = simpleGE.Button()
    self.btnPlay.text = "Play (Up)"
    self.btnPlay.center = (100, 400)
    self.btnQuit = simpleGE.Button()
    self.btnQuit.text = "Quit (Down)"
    self.btnQuit.center = (550, 400)
    self.sprites = [self.instruction, self.lblScore, self.btnPlay, self.btnQuit]

# Method to process the instruction scene logic
```

```
    def process(self):
        if self.btnQuit.clicked:
            self.response = "quit"
            self.stop()
        if self.btnPlay.clicked:
            self.response = "play"
            self.stop()
        if self.isKeyPressed(pygame.K_UP):
            self.response = "play"
            self.stop()
        if self.isKeyPressed(pygame.K_DOWN):
            self.response = "quit"
            self.stop()


# Define the main function
def main():
    # Initialize variables
    keepGoing = True
    score = 0
    # Main game loop
    while keepGoing:
        # Initialize instruction scene
        instruction = Instruction(score)
        instruction.start()
        # Check player response
        if instruction.response == "play":
            # Start the game scene
```

```
            game = Game()
            game.start()
            score = game.score
        else:
            # Exit the game loop
            keepGoing = False

# Run the main function if the script is executed directly
if __name__ == "__main__":
    main()
```

Future Improvements:

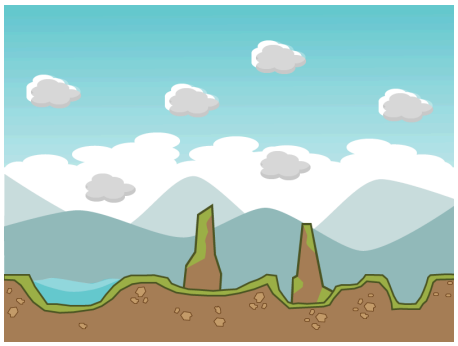
Add different levels with increasing difficulty.

- Introduce power-ups and obstacles to enhance gameplay variety.
- Implement high-score tracking and leaderboard functionality.
- Add visual effects and animations to improve the overall experience.
  Include power-ups or obstacles to enhance gameplay variety.
  Improve graphics and visual effects for a more immersive experience.

Conclusion:

The Coin Collector Game provides a simple yet engaging gaming experience suitable for players of all ages. With its intuitive controls, delightful sound effects, and challenging gameplay, it promises hours of entertainment and fun.

Game Assets:



Beach background from
https://opengameart.org/

Coin from
https://opengameart.org/

Boy character “Kross” from
https://opengameart.org/

Background Audio free open source from:
https://pixabay.com/music/search/genre/video%20games/

Coin sound free open source from
https://mixkit.co/free-sound-effects/coin/