Republic of the Philippines
Commission on Higher Education
University of the Philippines - Diliman
Diliman, Quezon City

College of Engineering
Computer Science 11

Unraveling the Secrets of Dropping:

A Programmer's Guide for the Game "SINGKO GAMING!"
Machine Problem 2

Carvajal, Ivan Rev
Cuaresma, Mark Lester
Moreno, Sofia Marie
GROUP 28
CS 11 Students

Edgardo Felizmenio
CS 11 Instructor

December 4, 2018
INTRODUCTION

"If you can code, you can create video games."

Video games like Clash of Clans, League of Legends, Halo and World of Warcraft are all products of complex programming that involves different algorithms and graphic interfaces that are beyond an ordinary person's scope of thinking. But these complicated codes that build up these games are all rooted to the same idea of programming behind simple games like Minesweeper and Pac-man. Without the basic principles that guide the programmers of this legacy games, all of the games in this generation will cease to exist.

The game SINGKO GAME! is a lightweight and simple game that allows the user/player to control a character using keyboard/mouse to hover around an area where random obstacles and bonuses emerge upon. Unlike the games that thrive the modern computers and gadgets, SINGKO GAMING! is composed of simple codes that makes use of basic principles of programming, such as list, functions, strings and files that are integrated to produce the game itself, while using basic Graphic User Interface (GUI) like Pyglet to animate the game and enhance the user experience. In this programmer's guide entitle "Unraveling the Secrets of Dropping", it aims to enlighten the readers on the insights behind its game engine and interface, the concepts followed throughout the worktime and justify how such simple games become the foundation of the development of more complex video games that exists and will exist in the unending technological revolution of this generation.

- The Programmers

THE GAME DESIGN

- *"All of our UP life, we try to avoid a grade of 5, without knowing that this was all an illusion, there is no escape from this downfall, we are on the midst of failing, but how long can we endure the challenge, how long till we officially dropped and surrender?" - Programmers*

I. Behind the SINGKOS

The program SINGO GAME! is a manifestation of the programmers' cumulative learning in their entire CS 11 journey which aims to create an application that is critically based on the concepts and skills that were tackled during the entire semester. The game is inspired by the current game applications that can be downloaded in smartphones, such as Rise Up and Smash, which have a similar goal for the user: an unlimited dodging experience that tests the player's ability to endure infinite waves of obstacles. This simple application may seem too conventional for the public, but what it makes unique is its ability to relate with the student's worst fear, failing grades. It is more than a series of dodging and catching falling debris, but rather, it is also a challenge for the fellow Isko/Iska of the creators that are striving to do their best, conquering all the odds and standing up to fight for their honor and excellence. #UPFight.

This program is a single-player game which is run in Python 3 interpreter, with the use of Pyglet (a GUI) to animate its interface and allows a more interactive environment for the player. The controls of the game can be accessed using directional keys in the keyboard or by the manipulation of the mouse cursor, depending on the user's preference. Scores can be attained throughout the game by catching the falling grades, with the exception that if the grade of singko has been caught by three times, the game is over. Specific points are allotted to each type of grade, that will add up until the player touches with the fiery doom of the singko. High scores are tracked to give the player something to beat during their entire gaming session, thus increasing the overall excitement.

II. MAIN GAME MODULE (`main.py`)

```
import pyglet
import engine

engine.game_proper()
engine.introduction()
```

This module is composed only of four lines of codes which imports pyglet, the game engine module and calling out the two function engine.game_proper() and engine.introduction() which initializes the game for the player to enjoy.

III. GAME ENGINE (`engine.py`)

The game engine is the core of the entire program, where it contains the principal algorithms and functions that will run the game. For the SINGKO GAMING!, Several defined functions are needed to construct the entire game engine. These algorithms serve different purposes that allow the continuity in the process included in the game. The following necessary codes are:

A. Importing Modules

```
import pyglet
from pyglet.gl import *
from random import randint
import datetime
import random
from pyglet.window import key
from pyglet.gl import*

def game_proper(): #This is what happens when the game is launch
    sound = pyglet.media.load('music/song.wav')
    looper = pyglet.media.SourceGroup(sound.audio_format, None)
    looper.loop = True
    looper.queue(sound)
    p = pyglet.media.Player()
    p.queue(looper)
    p.play()
```

The following codes imports the built-in modules that are present in the interpreter that are crucial in running the game. The random module allows for the randomization of where the falling grades will occur, the datetime for the timer, and the piglet.gl for the use of transparent PNGs.

B. Background Music

In order to run the game with a background music, this function plays the theme song of the game "Voltes V" once the game is launched, which gives more interactive experience to the player and a song the he/she can relate upon.

```
def game_proper(): #This is what happens when the game is launched. It also loads the sounds
    sound = pyglet.media.load('music/song.wav')
    looper = pyglet.media.SourceGroup(sound.audio_format, None)
    looper.loop = True
    looper.queue(sound)
    p = pyglet.media.Player()
    p.queue(looper)
    p.play()
```

C. Window & Images Setup

The following codes are responsible in setting up the window size and also the images that are used in the game, such as the background pictures, the avatar of the user, the falling grades, the antagonist wizard, and others that have significant purposes in the game:

```
window = pyglet.window.Window(640, 480, "Singko Gaming") #Size of the Window and the title
icon1 = pyglet.image.load('icons/cinco.png') #This is for the icon
window.set_icon(icon1)
megaman = pyglet.resource.image('icons/megaman.png') #This is the character sprite
megaman.anchor_x = megaman.width // 2
megaman.anchor_y = 0

game_bg = pyglet.resource.image('bg/game_opening_bg.jpg') #This is the background of the opening
game_bg.anchor_x = 0
game_bg.anchor_y = 0
#Below are the icons used for the corresponding grades
uno = pyglet.resource.image('icons/uno.png')
uno.anchor_x = uno.width // 2
uno.anchor_y = uno.height

dos = pyglet.resource.image('icons/dos.png')
dos.anchor_x = dos.width // 2
dos.anchor_y = dos.height

tres = pyglet.resource.image('icons/tres.png')
tres.anchor_x = tres.width // 2
tres.anchor_y = tres.height

cuatro = pyglet.resource.image('icons/cuatro.png')
cuatro.anchor_x = cuatro.width // 2
cuatro.anchor_y = cuatro.height

dropped = pyglet.resource.image('icons/dropped.png')
dropped.anchor_x = dropped.width // 2
dropped.anchor_y = dropped.height

cinco = pyglet.resource.image('icons/cinco.png')
cinco.anchor_x = cinco.width // 2
cinco.anchor_y = cinco.height
```

D. Pre-game Features

a. Introduction Function

This function presents the title of the game that will be immediately shown once the game is run. It contains the codes that will respond to the users input leading to function succeeding it.

```
#This is page when the player decided to continue
def introduction():

    introbg = pyglet.resource.image('bg/intro_bg.jpg')
    introbg.anchor_x = 0
    introbg.anchor_y = 0


    @window.event
    def on_draw():
        window.clear()
        introbg.blit(0, 0)

    @window.event
    def on_key_press(symbol, modifiers):
        if symbol == key.RIGHT:
            shortstory()
```

b. Short Story Function

On the other hand, this function acts the same as the previous one, except that it contains the introductory story that gives a statement in which the user can relate upon. It also accepts input from the player that will navigate towards the last two features of the pre-game

```
#This is the page when the player decided to continue
def shortstory():

    storybg = pyglet.resource.image('bg/story_bg.jpg')
    storybg.anchor_x = 0
    storybg.anchor_y = 0


    @window.event
    def on_draw():
        window.clear()
        storybg.blit(0, 0)

    @window.event
    def on_key_press(symbol, modifiers):
        if symbol == key.RIGHT:
            guidelines()
```

c. Guidelines Function

This function is used to give the user an idea on how to play the game which shows the rules that were needed to follow in order to attain a high score. Similar with the previous functions, it directs the game into the last part of the pre-game interface.

```
def guidelines():

    guidelinesbg = pyglet.resource.image('bg/guidelines_bg.jpg')
    guidelinesbg.anchor_x = 0
    guidelinesbg.anchor_y = 0


    @window.event
    def on_draw():
        window.clear()
        guidelinesbg.blit(0, 0)

    @window.event
    def on_key_press(symbol, modifiers):
        if symbol == key.RIGHT:
            alert()
```

d. Alert Function

```
#This is for alert purposes, if the player wants to go back to guidelines
def alert():

    alertbg = pyglet.resource.image('bg/alert_bg.jpg')
    alertbg.anchor_x = 0
    alertbg.anchor_y = 0


    @window.event
    def on_draw():
        window.clear()
        alertbg.blit(0, 0)

    @window.event
    def on_key_press(symbol, modifiers):
        if symbol == key.LEFT:
            guidelines()
        if symbol == key.RIGHT:
            maingame()
```

This function on the other hand gives the final direction to the user regarding on the rules of the game, on how to control the character and dodge the falling singko grades. After this function is executed, once the user hit the right key, it will then proceed to the game proper. The user also has the option to return to the guidelines by pressing the left key.

E. Main Game

The main game is subdivided into more functions that build of the entire main game function. Once this function is called, all the functions and commands included inside this are run throughout the game. The following codes included the main game are as follows:

a. Game Over Function

```
def game_over():
        game.finalscore.append(game.score)
        score=open("text/score.txt", "w+") #This is for the program to record the previous scor
        score.write(str(game.finalscore[0]))
        score.close
        highestscore=open("text/highestscore.txt", "r+") #This is for the program to record the
        highestscorel = highestscore.readlines()
        highestscorevalue = highestscorel[0]
        highestscore.close
        if int(highestscorevalue) < game.score: #Just to check if the present user has beat the
            highestscore=open("text/highestscore.txt", "w+")
            highestscore.write(str(game.score))
            highestscore.close
        gameoverbg = pyglet.resource.image('bg/game_over_bg.jpg')
        gameoverbg.anchor_x = 0
        gameoverbg.anchor_y = 0
        game_over_score_label.text = 'Game Over! Your final score is: %d' % game.finalscore[0]
        @window.event
        def on_draw():
            window.clear()
            gameoverbg.blit(0, 0)
            game_over_score_label.draw()
        @window.event
        def on_key_press(symbol, modifiers):
            if symbol == key.RIGHT:
                introduction()
```

This sub-function presents the final score of the player and updates the highest score and the last game score; a message that will taunt the user to play the game again would be shown. If the player decided to play once more and responded to the game, it will proceed to the introduction part of the program again.

b. Game Blueprint

```
class Game(object):
    def __init__(self):
        self.megaman_x = window.width // 2
        self.megaman_y = window.height // 2
        self.uno_x = random.randint(0, window.width) #This i
        self.uno_y = window.height
        self.dos_x = random.randint(0, window.width)#This is
        self.dos_y = window.height
        self.tres_x = random.randint(0, window.width)#This i
        self.tres_y = window.height
        self.cuatro_x = random.randint(0, window.width)#This
        self.cuatro_y = window.height
        self.dropped_x = random.randint(0, window.width)#This
        self.dropped_y = window.height
        self.cinco_x = random.randint(0, window.width)#This
        self.cinco_y = window.height
        self.score = 0
        self.unoscore = 0
        self.dosscore = 0
        self.tresscore = 0
        self.cuatroscore = 0
        self.droppedscore = 0
        self.cincoscore = 0
        self.finalscore = []
        valuescore=open("text/score.txt", "r+") #This is to
        valuescorel = valuescore.readlines()
        valuescorevalue = valuescorel[0]
        valuescore.close
        valuehighestscore=open("text/highestscore.txt", "r+"
        valuehighestscorel = valuehighestscore.readlines()
        valuehighestscorevalue = valuehighestscorel[0]
        valuehighestscore.close
        self.lastscore = int(valuescorevalue)
        self.highestscore = int(valuehighestscorevalue)
        self.gameover = False
```

This part of the game makes us of the other class types objects in order to add several features of the game such as the character, falling grades, the current score, the previous scores, and highest scores of the game. It includes an inner function inside it, that is usually used for new created objects.

c. Labels

The following codes use the Label widget class that allows the programmer to display and position text of images in the GUI that are encoded in the interpreter. It includes the game itself, the last score, highest score and game over score of the game.

```
#This is for printing the scores for the game
    score_label = pyglet.text.Label(str(game.score),
                                    x=0,
                                    y=window.height,
                                    anchor_x='left',
                                    anchor_y='top')
    unoscore_label = pyglet.text.Label(str(game.unoscore),
                                    x=66,
                                    y=366,
                                    anchor_x='left',
                                    anchor_y='top')
    dosscore_label = pyglet.text.Label(str(game.dosscore),
                                    x=66,
                                    y=318,
                                    anchor_x='left',
                                    anchor_y='top')
    tresscore_label = pyglet.text.Label(str(game.tresscore),
                                    x=66,
                                    y=270,
                                    anchor_x='left',
                                    anchor_y='top')
    cuatroscore_label = pyglet.text.Label(str(game.cuatroscore),
                                    x=66,
                                    y=222,
                                    anchor_x='left',
                                    anchor_y='top')
    droppedscore_label = pyglet.text.Label(str(game.droppedscore),
                                    x=66,
                                    y=174,
                                    anchor_x='left',
                                    anchor_y='top')
    cincoscore_label = pyglet.text.Label(str(game.cincoscore),
                                    x=66,
                                    y=126,
                                    anchor_x='left',
                                    anchor_y='top')

    lastscore_label = pyglet.text.Label(str(game.lastscore),
                                    x=window.width,
```

d. Control Functions

The following codes are responsible for the motion and input of the either keyboard or mouse which allows the player to move the character as it dodge and catches the falling grades. The user can either click the directional keys to go up, down, left, right or use the mouse to move the cursor and the character will follow the cursor.

```
#This is for windows event such as clicking the di
    @window.event
    def on_key_press(symbol, modifiers):
        if symbol == key.RIGHT:
            game.megaman_x += 10
        elif symbol == key.LEFT:
            game.megaman_x -= 10
        if symbol == key.UP:
            game.megaman_y += 10
        elif symbol == key.DOWN:
            game.megaman_y -= 10
        elif symbol == pyglet.window.key.ESCAPE:
            window.close()
    @window.event
    def on_mouse_motion(x,y,dx,dy):
        game.megaman_x = x
        game.megaman_y = y
    @window.event
```

e. Screen When Not Game Over Function

```
def on_draw():
    if not game.gameover:
        window.clear()
        glEnable(GL_BLEND)
        game_bg.blit(0, 0)
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)#This is basically to make t
        megaman.blit(game.megaman_x, game.megaman_y)
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)#This is basically to make t
        uno.blit(game.uno_x, game.uno_y)
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)#This is basically to make t
        dos.blit(game.dos_x, game.dos_y)
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)#This is basically to make t
        tres.blit(game.tres_x, game.tres_y)
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)#This is basically to make t
        cuatro.blit(game.cuatro_x, game.cuatro_y)
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)#This is basically to make t
        dropped.blit(game.dropped_x, game.dropped_y)
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)#This is basically to make t
        cinco.blit(game.cinco_x, game.cinco_y)
        score_label.text = 'Your score is: %d' % game.score
        score_label.draw()
        unoscore_label.text = ' %d' % game.unoscore
        unoscore_label.draw()
        dosscore_label.text = ' %d' % game.dosscore
        dosscore_label.draw()
        tresscore_label.text = ' %d' % game.tresscore
        tresscore_label.draw()
        cuatroscore_label.text = ' %d' % game.cuatroscore
        cuatroscore_label.draw()
        droppedscore_label.text = ' %d' % game.droppedscore
        droppedscore_label.draw()
        cincoscore_label.text = ' %d' % game.cincoscore
        cincoscore_label.draw()
        lastscore_label.text = 'Last Game Score: %d' % game.lastscore
        lastscore_label.draw()
        highestscore_label.text = 'Highest Game Score Ever: %d' % game.highestscore
        highestscore_label.draw()
```

This function uses the glBlend function to make the PNG images transparent and blend with the game. It removes the unnecessary backgrounds in the images giving a smoother texture of the entire game. The function also adds the string texts that describes the different scores stored in the memory of the interpreter.

f. Ball Drop Functions

The ball_drop function serves two purposes, first, it is responsible for what will happen in the collision of falling grades and the character. If the singko grade has been crossed thrice, the game will be over, or else it will append the assigned value of the grade in the score it will attain. It updates the score of the game by interpreting the interaction between the character and the game. On the other hand, it is also responsible for continuous falling grades that appear randomly in the screen by using the random module that was opened in the start of the engine.

```
def grades_drop(dt):
    #This is for collission.
    if game.cincoscore == 3: #If the condition that the cinco has been cros:
        game.gameover = True
        return
    if (abs(game.cinco_x - game.megaman_x) < 30 and abs(game.cinco_y -
            game.megaman_y) < 30):
        game.cincoscore += 1
        game.cinco_x = random.randint(0, window.width)
        game.cinco_y = window.height
    if (abs(game.uno_x - game.megaman_x) < 30  and abs(game.uno_y -
            game.megaman_y) < 30):
        game.score += 1
        game.unoscore += 1
        game.uno_x = random.randint(0, window.width)
        game.uno_y = window.height
    if (abs(game.dos_x - game.megaman_x) < 30 and abs(game.dos_y -
            game.megaman_y) < 30):
        game.score += 2
        game.dosscore += 1
        game.dos_x = random.randint(0, window.width)
        game.dos_y = window.height
    if (abs(game.tres_x - game.megaman_x) < 30 and abs(game.tres_y -
            game.megaman_y) < 30):
        game.score += 3
        game.tresscore += 1
        game.tres_x = random.randint(0, window.width)
        game.tres_y = window.height
    if (abs(game.cuatro_x - game.megaman_x) < 30 and abs(game.cuatro_y -
            game.megaman_y) < 30):
        game.score += 4
        game.cuatroscore += 1
        game.cuatro_x = random.randint(0, window.width)
        game.cuatro_y = window.height
    if (abs(game.dropped_x - game.megaman_x) < 30 and abs(game.dropped_y -
            game.megaman_y) < 30):
        game.droppedscore += 1
        game.dropped_x = random.randint(0, window.width)
        game.dropped_y = window.height
```

```
#This code is for resetting the ball.
if game.uno_y < 0:
    game.uno_x = random.randint(0, window.width)
    game.uno_y = window.height
else:
    game.uno_y -= 2
if game.dos_y <0:
    game.dos_x = random.randint(0, window.width)
    game.dos_y = window.height
else:
    game.dos_y -=2
if game.tres_y <0:
    game.tres_x = random.randint(0, window.width)
    game.tres_y = window.height
else:
    game.tres_y -=2
if game.cuatro_y <0:
    game.cuatro_x = random.randint(0, window.width)
    game.cuatro_y = window.height
else:
    game.cuatro_y -=2
if game.dropped_y <0:
    game.dropped_x = random.randint(0, window.width)
    game.dropped_y = window.height
else:
    game.dropped_y -=2
if game.cinco_y <0:
    game.cinco_x = random.randint(0, window.width)
    game.cinco_y = window.height
else:
    game.cinco_y -=2
```

Timer

```
pyglet.clock.schedule(grades_drop)#This is to know when the grades will dropp
```

IV. MODULARIZATION

The game contains the following:

FOLDERS

bg

It contains all the backgrounds used in the game.

icons

It contains all the icons such as the grade icons as well as the character sprite.

music

It contains the music played in the game.

test

It contains two files:

highestscore.txt – This is where the highest ever score recorded is contained.

score – This is where the score of the last game is recorded.

FILES

engine.py

It contains the core algorithm of the game.

main.py

It is the file that user opens when he/she wants to play the game.

Programmer's Manual.pdf

This is what the user opens if he/she wants to know how the game has been created.

V. REFERENCES

The programmers are indebted to the following:

PEOPLE

Professor Edgar Felizmenio

BOOKS

Think Python 2: How to Think Like A Computer Scientist 2nd Edition

CODES

https://github.com/dawran6/pyglet-tutorial/blob/master/megaman.py

In this short Pyglet game, we took inspiration (but our game is way different, our gaving having more features and codes) for our game. The megaman sprite was also borrowed from this.

https://stackoverflow.com/questions/46044714/pyglet-loading-blitting-image-with-alpha

This is where we got our code for making the PNG transparent files appear transparent in the game.

https://stackoverflow.com/questions/27391240/how-to-play-music-continuously-in-pyglet

This is where we got out for making sure that the song will loop continuously.

IMAGES

Clouds : ***https://www.kisspng.com/png-cloud-clip-art-clouds-clipart-388777/****i*

You Shall Not Pass: ***https://www.amazon.com/Gandalf-ShallStickerBumperWindow/dp/B01N2U0OA8***

Fire: ***https://www.pinterest.ph/pin/398639004487716905/***

SOUNDS

Voltes V soundtrack

VI. SAMPLE GAME INTERFACE