Media Computing Project

Python 3 blackjack game

Jose Flores, Andrew Long
Computer Science
University of Massachusetts Lowell
Lowell, MA United States of America

***Abstract*—To produce a Python 3 blackjack simulator with a pyGame generated user interface in lieu of completing any other assignment or exams with the intent of fulfilling the requirements for COMP 1000 Media Computing under professor Byung Kim for the Fall 2016 semester.**

Keywords—Python 3, pyGame, GUI, blackjack, Media Computing

I. Project Expectations

A. Guidelines

- All contributions to the project will be stored in Github.
  - The *master* branch of the *blackjack* repository of the *uml-cm-f16* organization will hold the application Ref. [1].
  - Documentation shall be found on the *gh-pages* branch of the *blackjack* repository of the *uml-cm-f16* organization Ref. [2].
  - Documentation shall be displayed on the project Github webpage Ref. [3].
- Unit tests will be written for any element that is not part of the Graphical User Interface (GUI).
- Usability tests will be carried out for the any element that is part of the GUI

B. Functionality

- The project will have an installer to setup the project and any of its dependencies, as well as carry out any application/ project actions.
- The simulator will allow for gameplay between one human player and a computerized dealer.
- The simulator will follow the rules as described in the bicycle cards webpage Ref [4].
- The simulator will show the probability to win a hand, taking into account the cards in play to the screen.

II. Blackjack Simulator

The blackjack simulator will be broken up into two parts, a fully functional blackjack terminal simulator and a GUI interface.

A. Terminal Simulator

Will allow for a hand of blackjack to be played and display the appropriate output. Shows cards being flipped and prompts the user for action. During a human players turn it will display the probability of them winning the hand. The simulator will automate a dealers turn and determine who won the hand once the dealer stops play.

B. Graphical User Interface

A pyGame GUI will be developed to allow clickable play. This GUI will intercept the commands from the UI and pass them to the terminal simulator classes, retrieving output and translating them into a displayable action.

The GUI will not be unit tested, but it will be put through a usability test. The test will verify that all elements work as intended and that consecutive actions are non-blocking.

C. Task Ownership

In the following table ownership of a task is assigned by a single letter system. The letter *A* represents that a task has been assigned to *Andrew Long*, and the letter *J* represents that a task has been assigned to *Jose Flores*. The letter *B* signifies that *both authors* have been assigned this task and is used in research stages and final testing.

TABLE I. TASK OWNERSHIP

Task	**Description**	**Owner**
Python 3	Learn about Python 3.	B
PyGame	Learn about PyGame.	B
Sphinx	Learn about Sphinx documentation.	B
Setup.py	Python script to install dependencies, generate documentation, and deploy to Git branches.	J
Card.py	A card class.	J
Card.test.py	Unit tests for the card class.	J
Deck.py	A deck class of 52 cards.	J

Task	Description	Owner
Deck.test.py	Unit tests for the deck class.	J
Hand.py	A hand of cards class.	J
Hand.test.py	Unit tests for the hand class.	J
Player.py	Extends Hand by giving player actions.	J
Player.test.py	Unit tests for the player class.	J
Dealer.py	Extends Player by giving deck handling actions.	J
Dealer.test.py	Unit tests for the dealer class.	J
Blackjack.py	Extends Dealer by adding blackjack rules	J
Blackjack.test.py	Unit tests for the blackjack class.	J
Probability.py	A Probability class. Allows for the player's chance to win to be calculated	J
Probability.test.py	Unit tests for the probability class.	J
Game images	Card, buttons, background, etc.	A

Task	Description	Owner
UI design	Determine layout, calculate positions for buttons graphics and probability display.	A
UI proof of concept	Create a window, add UI elements, and input elements. Test inputs and output functionality.	A
UI game flow	Make sure game actions can be implemented. Such as adding a card, and folding for both the dealer and player	A
UI/ Sim integration	Integrate UI to call simulator	A
Game testing	Test completed application for functionality and usability.	B

References

[1] *https://github.com/uml-cm-f16/blackjack.*

[2] *https://github.com/uml-cm-f16/blackjack/tree/gh-pages*

[3] *https://uml-cm-f16.github.io/blackjack/*

[4] http:// *www.bicyclecards.com/how-to-play/blackjack/*