Darien Flamont

Rastin Piri

6 April 2019

CMPT 365 – Final Project Report

***Introduction***

Our project is a blackjack video game that includes an opening title sequence animation accompanied by music and sound effects when interacting with the blackjack board. This game was created using pygame and the python coding language [1] [2]. Using card images found on the internet we were able to convert the images to portable network graphics (i.e. PNGs) [6] and use them to create sprites that move around the screen and disappear/reappear based on the game state. Our welcome screen logo was made by us along with the interactive buttons used for playing the game, the background and cards [6] [7] were found online but we had to make the transparent alpha layer so they showed up better against our felt background. We also found many casino game sounds on the internet that we used when transitioning between different game states so we can give the player a sense of satisfaction. Along with the gambling aspect of the game we thought it was a very addictive game with good interactive elements and lots of multimedia aspects, we have played it many times in testing and for fun.

***Project Description***

Our Blackjack written entirely in Python with the library pygame. We used royalty free sprites found online that include all 52 cards found in a normal deck. The premise of the game is exactly like that of blackjack but a simplified version based on the typical Casino setting. The player will be greeted with a title screen that plays a nice animation of the title of the game “blackjack” traveling into the middle of the screen with a vibrant gradient color attached to the logo, background, and play button [2]. After clicking the play button with the mouse, the player will be able to play multiple rounds of blackjack with starting funds of 500 dollars. The player then will be able to interactively play a 1v1 blackjack game against an AI dealer, the dealer will hit if they have less than 17 and will try to beat the player and deplete their funds. The player has 5 buttons to choose from deal, hit, stay, betup and betdown and they are clickable by using the left mouse button (betup and betdown also have the ability to increase the bet when holding the mouse button down). These are the basic blackjack options and control the flow of the game. The deal button is only accessible to the player when a round has not yet started and when the “deal” button is clicked it will invoke the functions that will initialize and reshuffle the deck and also deals 1 face down card and 1 face up card to the dealer and 2 cards face up to the player. One interesting implementation we used was using modulus calculations so that the player is dealt the first card, the dealer is dealt the second, and so forth alternating between player and dealer, adding more realistic game logic. After this deal the hit and stay buttons are activated and available for use. The hit button will allow the player to be dealt subsequent cards from the “dealt” deck before the dealers turn. If the player has less than 21 they are allowed to continuously hit until the stay button is clicked or the player “busts” (gets over 21). When the player stays it is now the dealers turn; the dealer will flip their face down card and hit if they have less than 17. If the dealer busts the player wins if the dealer does not then the scores are compared (more on this below). This cycle is repeated with a freshly initialized deck at the beginning of each round. At the beginning of each round the player will be able to set their bet using the wooden arrow buttons before hitting the deal button. After the deal button is invoked the associated bet the player chose will be subtracted from their funds and the betting arrows will stay on the screen but will not produce any actions when clicked. The player is not allowed to continue playing once their funds are depleted and if the initial bet is higher than their funds we make sure the max bet is then changed to the maximum amount of funds the player has (i.e. if the player has $75 and they place a $50 bet and lose the subsequent bet will be changed to $25, because that is the maximum funds the player can use to play and they will not be able to bet above $25 but they can bet lower). If the player hits a true blackjack, which can only happen on the deal with an ace and either a 10 or facecard, the player will be paid out 3:1 so they will receive their initial bet back plus 2\*bet. If the player does not hit blackjack on the deal they will have the option to hit or stay as said above. If the player beats the dealer they will be paid out 2:1 so double their initial bet. The player or the dealer lose if they bust and in any scenario where the player and dealer have the same total (this includes blackjack) the player and dealer “push” which means they will get their bet back but win no extra money.

**Casino Blackjack Differences:** The three factors that are included in **some** casino blackjack games but not our game of blackjack are “insurance”, “double down”, and “splitting” [5]. Insurance is a rule in **some** casinos that allows the player to put a side bet down if the dealer is showing a face card or an ace. This allows the player to win 2:1 odds on a side pot if the dealer does not have blackjack after the dealer hand is checked on the deal round. We did not implement this because we wanted the original thrill of blackjack to thrive without extra bets and side pots for new players to learn; we wanted to implement the simplest blackjack experience that still gives the player the satisfaction of seeing their cards on the screen versus the dealers and betting money to win. The second house rule we did not include was double down. This rule would have been easy to implement and ups the initial bet by 2, but the double down function can only be used on the dealt hand (i.e. player and dealer have 2 cards each) so that they can double their bet if they had really good cards. This again was left out for simplicity sake and the sole purpose that most novice players of blackjack would not know the rules of doubling down and be frustrated with a function that is only being used at some points in the game. We also did not include the option to split the cards which means that if we were dealt two cards of the same value we would be able to make two new hands that are independent and able to hit on both. This was excluded again for simplicity and for the fact we just did not have room on our playing board to implement multiple hand sprites.

**Multimedia Elements:** Video Games in general have most if not all of the multimedia elements seen in class. From sound effects, interactive buttons, and the ability to see changes appear in real time, video games create a visceral experience for the player and keeps them coming back for more. Our blackjack game is no different. The initial screen has some nice Casino music player that will continue to play while the player is at the blackjack table. The title screen also has an animation created by us that has the name of the game appear from the left screen and center itself. After the title screen the player is created with a shuffling sound effect to introduce them to the blackjack board and has a nice felt background along with buttons for betting, deal, and a facedown deck for aesthetic purposes. The player also has valuable information given to them right away with their funds and bet totals sitting between the bet buttons along with helpful prompts. After clicking and interacting with the buttons; funds will be depleted, cards will be dealt, and new buttons will appear giving the player interactiveness and change with the initial round and every round after that. Some other notes on multimedia elements is that we wanted the title screen to be flashy and show our creative side with the gradient effect and cool logo but the blackjack table itself is more traditional in both color and casino elements outside of our title screen. We chose wooden buttons for our betting as wood is synonymous with felt tables and chose a card backing that is very traditional along with the tradition sets of cards that are displayed for the player. On top of that for our “deal”, “stay”, and “hit” buttons we used a color picker and color wheel on the face cards of our deck and implemented those exact colors (gold, red, and royal blue) for our buttons so that continuity was kept between the card colors and the environment the player is submersed in. We also used a an old western style font [4] for our buttons to keep that environment fresh as well. Instead of being flashy and super artistic with our game we decided to go with a more traditional approach and it really paid off in the end as the final product is very nostalgic looking but fresh at the same time.

**Issues:** The one issue we had when creating this game was handling the resources path. Both partners working on the game were using Windows OS so we were able to just hardcode our resource paths as a global variable at the top of our python files that used the resources path. We tried to use sys.join() which is a built in python function that will use the home directory on any operating and join it to our resource folder but for some reason this did not work on our OS’s and the IDEs we were using. This game may have a bit of problem when trying to run on a linux environment for TA testing purposes but will hopefully be self explanatory with embedded comments in the files. Darien got it working on linux once but afterwards the resources path stopped working on linux and the continued development was done on windows. It was also presented on a windows OS so hopefully the TA or Professor has a windows environment and the path can be changed to wherever the source code is located to play the game by running python3 PlayGame.py.

***Experimental Results***

**Proposal Process:** When proposing ideas for our project we originally thought of maybe doing a travel website for Vancouver that would show tourist information such as food recommendations, local music, and implementing the Google maps API to find hotels and parks nearby. We also proposed using motion tracking devices such as the Leap Motion to create an interactive pong game that could track hands and use them as paddles. Though this was an interesting idea the sdk was hard to understand and we were not able to get our hands on the Leap Motion for a good price so this idea was quickly changed into creating some sort of video game. We decided that since we used python for our assignment 2 we should try and use python again as we were somewhat comfortable with it now and knew there was an integrated library pygame that would help us with creating animated sprites and interactive buttons on the screen.

**First Iteration of our Video Game:** During the first iterations and when getting out of the proposal phase we decided that we wanted to do a Casino game with random chance and scoring being the amount of money that the player is betting on a hand. During the first iterations and after deciding on a Casino game we initially wanted to do multiple games encapsulated within a main menu that would be able to transition you between games and a global money count. It was based on the pygame community that helped develop the application “Pyroller” that has many different games such as keno, blackjack, bingo, etc. all encased within a global application that keeps track of your players progress [8]. We wanted to mimic this idea but with a limited subset of games originally slots, keno, and blackjack. After looking at the pygame documentation and all the different elements that go into making even one of these video games we soon realized we had to reign in our vision a bit and focus on a singular game and get it done right with proper animations and game mechanics. This lead us into creating our blackjack video game that still has a title screen and animations before entering the main game.

**Blackjack First Iteration:** The first iteration of our blackjack game was very rough and was missing many elements that are included in the final game. Firstly we did not have a funds and bet total appear on the screen when the game was loaded. Secondly the buttons were a static element that was always present on the game screen when at the blackjack table. Though this had been changed in future iterations it was nice to have all the colours of the buttons appear on the screen at one time so the player could see the continuity of the colour of the button to the colours on the face cards. This iteration did not include sound queues on certain buttons and when entering the game but did contain the intro music that followed into the blackjack table.

**Blackjack Second Iteration:**  Moving forward it wasn’t obvious which buttons were supposed to be played at what time during the round. Instead of creating a helper overlay or tutorial on how to play we decided to have the buttons appear and disappear on the screen in real time as the rounds end and begin to coax the player into not being able to click things that would otherwise break the blackjack experience and lose them money. We also included a screen total of the funds and bets that are nested between the betting arrows. This iteration was also the first iteration to include the bet buttons becoming unusable during a hand of blackjack and the bet was not manipulatable until the round was over. This iteration did have both players receiving cards but both of the dealers cards were facing upwards so we knew in our final game iteration to stay true to blackjack we wanted to include the facedown card that would only be flipped when the dealers turn was invoked. This iteration started to include some of our sound effects but not all, we had the opening sound included in this iteration but no button sound effects.

**Black Final Game:** The final game of blackjack is well what you see when playing the game. It includes all the elements talked about in previous iterations plus the fact we have the first dealer card face down until the dealers turn is invoked or they hit a blackjack off the deal. It also includes all sounds effects including the intro shuffle, the round beginning deal to player and dealer, and the card sound effects on hit and stay. It also includes hold down ability for betting so instead of constantly having to click the bet up or down function you can now instead hold the button and it will increase/decrease the bet as long as the mouse button is pressed down and you are over the button. This version has onscreen prompts that will tell the player how to interact with the game while it goes on and giving the player useful information such as how much they lost, how much the player won, and why they lost/won including “push”.

Note: the onscreen prompt for the player winning and how much they won includes the initial bet in the prompt calculation, we did this because the amount is removed on deal so to avoid confusion we include the initial bet they placed in “winnings”. i.e. If the player bet 50 dollars and wins 2:1 the player technically only “won” 50 dollars and keeps the initial bet but our prompt says “Player won $100” to just show the player exactly how much is being deposited back into the funds. The math is all correct for winning and blackjack.

**Future Goals:** When thinking about features that can be included in this game in the future a couple immediately come to mind that have been subtly talked about in this report but not addressed fully. Firstly the game should be encapsulated as an executable file independent of python, that can just be downloaded and run on any system. Though we ran out of time to include this feature it would likely be fairly straight forward to pack this into a folder and run it as an executable. Secondly the fact we do not have a tutorial for our game can lead to new players being confused and not know what they are doing. A beginning tutorial would help the player understand the functionality of the game and allow it to be more complex. With the help of a tutorial window we would be able to support double down, insurance function, and splitting functions in the game because the player would be able to see in real time how the functions work and would be able to be walked through an example of each button. But in the last iteration with both the onscreen prompts and buttons becoming unclickable/disappearing we believe it is enough to help new players walk through their first couple rounds of our simple blackjack.

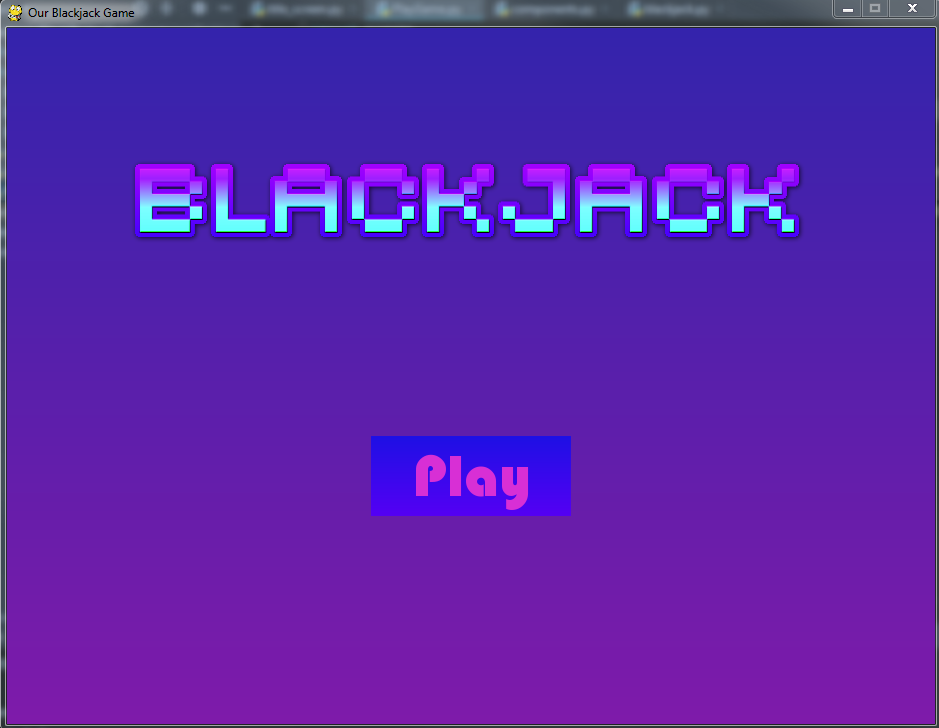
***Conclusion***

This game was a very fun project to work on and left us both with great satisfaction once it was completed. Though it looks like a simple game it took a very long time to code and has hundreds of lines of code to get all edge cases and options done for the player to have a full blackjack experience. We both learnt how hard it really is to animate different buttons and sprites in pure code even using the pygame library and python being known as a “high-level” language it took a very long time to be comfortable with the pygame classes and objects and how they interacted with the screen or “scene” as they are called. As well as the constant update based on tickrate had us solving many bugs because the game is being updated multiple times a second based on tickrate. This project left us with a very well rounded experience in video game making from tailoring the sprites to fit on screen and have transparent backgrounds, to game design and fitting all the elements in a neat fashion, and the actual game logic itself that is hiding underneath the hood of every action the player takes.

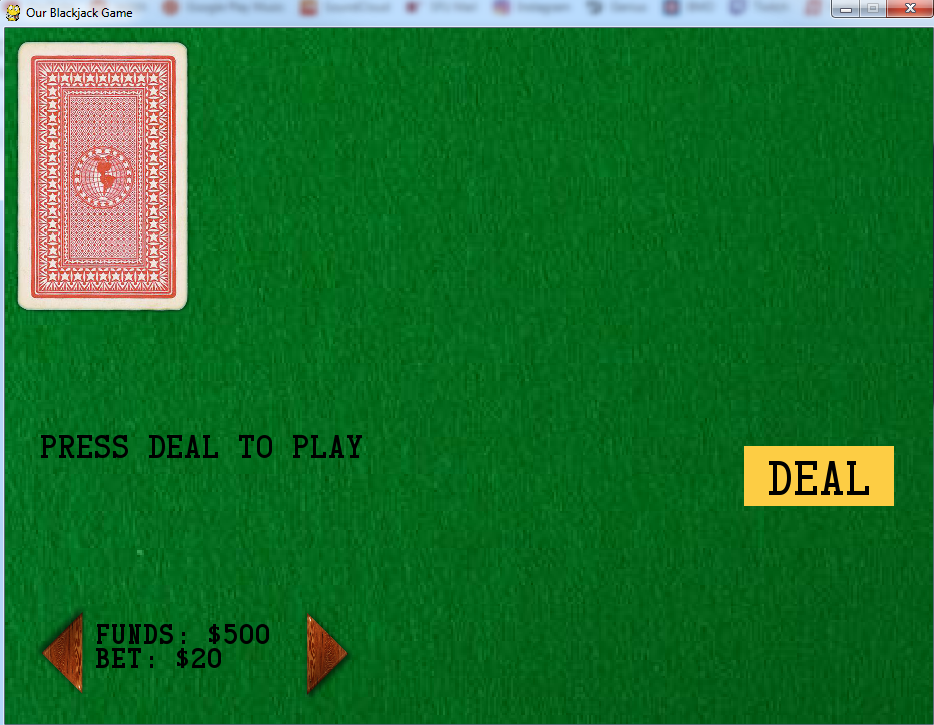
Overall this was a very fun project and am glad we chose a game and not just a front end superficial website that can be created in a day or two. This gave us a real challenge and we hope the hard work we put into this game is translated to a fun experience for anyone who plays the game.

***Screenshots***

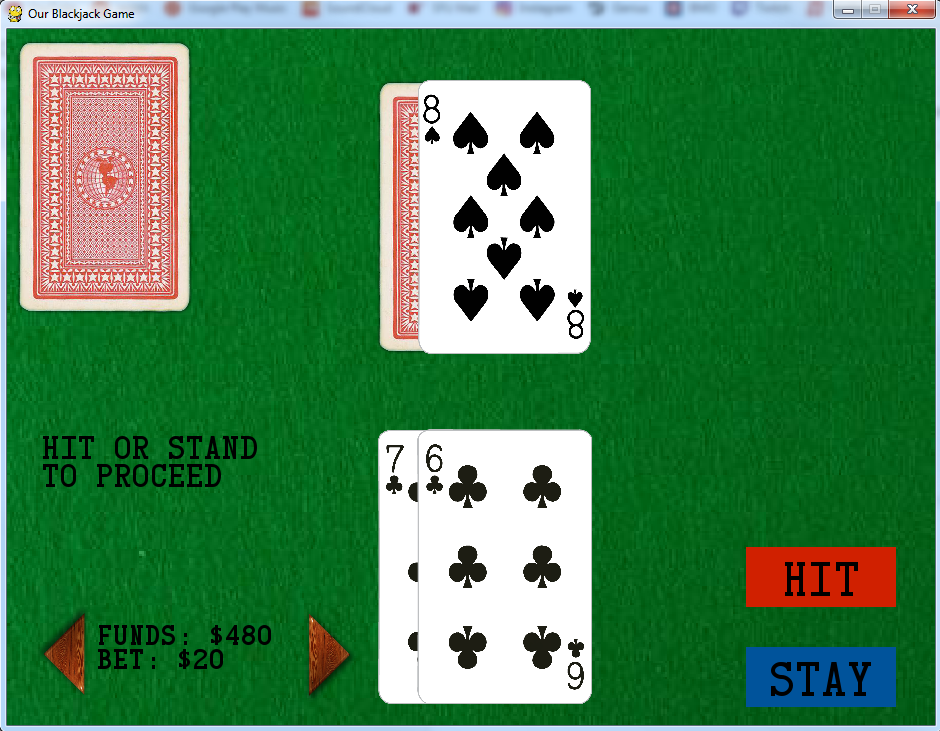
**Intro Screen**



**Beginning Game Screen**



**Deal Screen**

****

**Hit Screen**



**Dealers Turn Screen**

******

**Gameover Screen**

******

**References**

[1] pygame community, et al. (2019) *pygame documentation* [Online]. Available: <https://www.pygame.org/docs/> [Accessed 4 Apr. 2019].

[2] pygame community, et al. (2019) *Gradient Code* [Online]. Available: <http://www.pygame.org/wiki/GradientCode> [Accessed Apr. 2019].

[3] flummox3d (2014) *Blackjack.* [Online] sourceforge. Available: <https://sourceforge.net/projects/flumblackjack/> [Accessed 4 Apr. 2019].

[4] (2016) *Card Characters.* [Online] urbanfonts. Available: <https://www.urbanfonts.com/fonts/Card_Characters.font> [Accessed Apr. 2019].

[5] (2016) *Blackjack Rules.* [Online] pagat. Available: <https://www.pagat.com/banking/blackjack.html> [Accessed Apr. 2019].

[6] American Contract Bridge League. (2019) *52 Playing Cards.* [Online] Resource Center. Available: <http://acbl.mybigcommerce.com/52-playing-cards/> [Accessed Apr. 2019].

[7] salamatik. (2019) *Stock Photo - Green texture. Felt background for poker and casino table.* [Online] 123RF. Available: h[ttps://www.123rf.com/photo\_97519237\_green-texture-felt-background-for-poker-and-casino-table-.html](https://www.123rf.com/photo_97519237_green-texture-felt-background-for-poker-and-casino-table-.html) [Accessed Apr. 2019].

[8] Github community, et al. (2019) *pygrame casino* [Online] Github pyroller Available: <https://github.com/iminurnamez/pyroller> [Accessed Apr. 2019].