# Race to a Dollar Game

By Kush Patel

There are so many little kids in the world, they need to learn how to count, they need to know what money is, and they need to know how many coins equals 1 dollar, Race To a Dollar is a game in which two players can play, they pick any coin, they can't add any coins if the total would exceed one dollar, you put the coin in your collection if there is room in your collection. Another scenario is when you are buying money and do not have a credit card, and you need one more dollar. You don't have a dollar bill, you can count up for coins until it equals a dollar bill. You don't have a choice to pick an exact coin, you can only pick a valid coin, it is a rule that is made, the valid coins will be placed in front of you and you have to pick which coin is valid, a player will determine what is a valid move depending on the difference between your current amount and your target amount.The player is given the valid coins and decides what coin to use. The game will allow a player to pick a coin for 5 seconds, and if they exceed that amount then they automatically get chosen a random coin. The move is that you can pick a coin, but you can choose certain coins depending on how far the person is from the target. It doesn't randomly pick all coins, and the options for the coins narrow as they get closer to the target. This part shows that the strategy and goal of the game is to reach the exact target value not above or below, the player then strategises by checking to see if the other coins can be validly added and if they can be validly added then they'll pick that coin to add, to the collection, if the coin can't be added to the collection have to pick another type of coin, first one who reaches exactly one dollar wins. A human player will play against an AI player, and the human player will use input, whereas the AI player will use an algorithm that is provided in the textbook. This is a deterministic game because they know what the outcome will be if they pick a certain coin and dump it into their jar. They know that there are 4 different coins, each coin being a different value and if they add it to their current board, they will know the result of future actions depending on what the coin they pick. If it results in a valid result, they put it in their jar; otherwise, they would have to pick a different coin. Only one action is when you can pick a coin, but you can not look and pick a coin of your choice; it is still deterministic because you will know the outcome regardless of what coin you pick. At the end of the game, it'll display who won and the number of each type of coin they ended up with, like 1 nickel, 3 quarters, and 2 dimes. You just pick a coin that is valid enough to satisfy certain conditions, you don't pick a random coin, you only pick any of the valid coins. The game is designed to prevent the player from reaching an amount that is above the target, which is one dollar. It is also designed to make sure they reach the exact target amount, not below or above or else the game will never stop. If your current amount is 0.99, the only option you have is to pick only one penny, nothing else. This race to a dollar game is my version of it, with some rule changes in place, such as options will be narrowed as you get close to the target, and you can't exceed the target. The programming language I will be using is Python Programming language, it will also use a numpy library. There will be no datasets required since I'll be programming a game. There is existing code which are the algorithms and Game class code provided from the textbook, and you provided for programming assignment 2. The existing algorithm I will be using is an alpha beta pruning algorithm, in which player 1 will be the AI Agent, for the game rules, there'll be an algorithm in which a person chooses whatever coin they want to put in the game and the algorithm will check if that coin is valid or not and if that coin isn't valid to put in, they have to pick another coin to put into their pile, the algorithm makes sures if it is a valid action or not, if it isn't a valid action it'll force the player to put a new action and if they don't it'lll keep asking until they do so. I will slightly modify the alpha beta algorithm code to include a depth limit tracker to make sure it can calculate small values and prevent it from infinitely searching. I will modify the list of actions in the alpha-beta search by sorting them in descending order inside the alpha-beta search algorithm. Hence, it strategically makes a decision, I do not plan on changing the algorithm completely, just slightly modifying it to adapt to my current game environment. There'll be another algorithm that will detect the difference between the player's current amount and the target amount and that algorithm will generate a list of valid actions a player can put. The alpha beta pruning algorithm may choose a small number, which would take a long time for the code to reach, which would cause the search algorithm to run for a long time, so I added a depth tracker, which tracks the number of times it searched. If it reaches a certain depth limit, it stops searching because the AI algorithm needs to stop eventually or make a decision. I will read through the algorithm provided in class to see if it needs any work or modifications to adapt to my current algorithm code. The approach is that you can only add coins that are less than or equal to the target amount, it does a pre check to see if it exceeds the target amount, and if it does, then it'll ask you to input a another option and will keep on doing it until you do so, there'll be a utility in which it checks to see if the player has reached the target and if they did then they set the utility equal to 1 or -1 depending on who the player is and then there is a terminal test which determines when to end the game, utility checks to see when the game has ended or not by checking if it met the target amount and if it did, then the game terminates. An algorithm checks to see if the player on the board met the requirement, and if they did, it changes the end state. For the GUI, when it's the human player's turn, they have to click the make move button in the GUI and then enter their move in the terminal because the query player function is being called. For my GUI, I will be using the Tkinter Python library, which generates a graphical user interface. I will include the GUI in a class called GameGUI to make sure that the game only runs when the GUI is running. The buttons to make a move will only appear when it's the human player's turn; if it is the AI player's turn, the buttons won't appear. The GUI will track each player's current amount, along with all the coins in its list.  The GUI will show who is currently playing, the AI or the Human. The GUI will dynamically and consistently update the display of the game based on the current game state. For the first week, I will gather and analyse the requirements needed for this game, making sure that these requirements are valid, For the Second Week  I will then design the game with all of its rules and requirements, I will then read and analyse the algorithms that are provided in this class to determine what it is needed for this project. For the third week, I will start to write the code for this project. I will be writing this code in Python, The code will include all the rules to the game that will be created, and the algorithms for determining any valid or future actions will be coded. For the fourth week, I will test the project to see if all the features work. If there are any errors, I will debug them.My approach to debugging is to I will include print statements to check where the bug is occurring and what the bug is, and if there is a bug, I will modify the code around it after determining what the bug is and where in the code it needs some debugging. For the fifth week, I will create a GUI or any UI for this project.The GUI will show both players and the amount of money each player currently has and who the current player playing the game right now, it will also show the list of coins it has. There will be no special computing platform I will use. My GUI class will be responsible for starting the game. The classes that this project is related to were old classes I already took as an undergrad, like Python Programming(CPSC 223P). I will use the search algorithms from CPSC 481 as my AI agent when playing the game. I will also use the principles I learned from my Object Oriented Programming( CPSC 121) to help me construct my Game GUI class by including a constructor for the GUI, which would then create an Object for the Game GUI. The Object will directly run the game instead of calling certain functions; the functions in the Class will be inside the constructor. I will learn and look at any tutorials on how to use the Tkinker GUI library because I have never used the Tkinker GUI library in my past projects. I will watch some YouTube Videos or the GeeksForGeeks website on how to use the Tkinkter library. I am working alone on this Project, so I will be assuming all the roles in this project, I will be doing all the coding in this Project.