Team 19
Members: Jesse DeBok, Jack Ford, Ben Schulte, Caden LeCluyse, Nathan Bui
Initial Architecture Document

Ro-board

Project Synopsis:

An application for the Nao robot to act as a dealer or player in a game of our choosing.

Design:

As the market for digital entertainment grows, consumers demand a novel approach to playing physical games which better emulates the digital market (Figure 1). The idea of playing a non-human in a game has mostly been reserved to the digital market, but with Ro-board we are combining the digital and the physical. We are giving people the opportunity to bring their games out of obsolescence and into the modern age. By developing an application for the Nao robot, we can meet this demand and show there is room for both physical and digital entertainment in modern society.

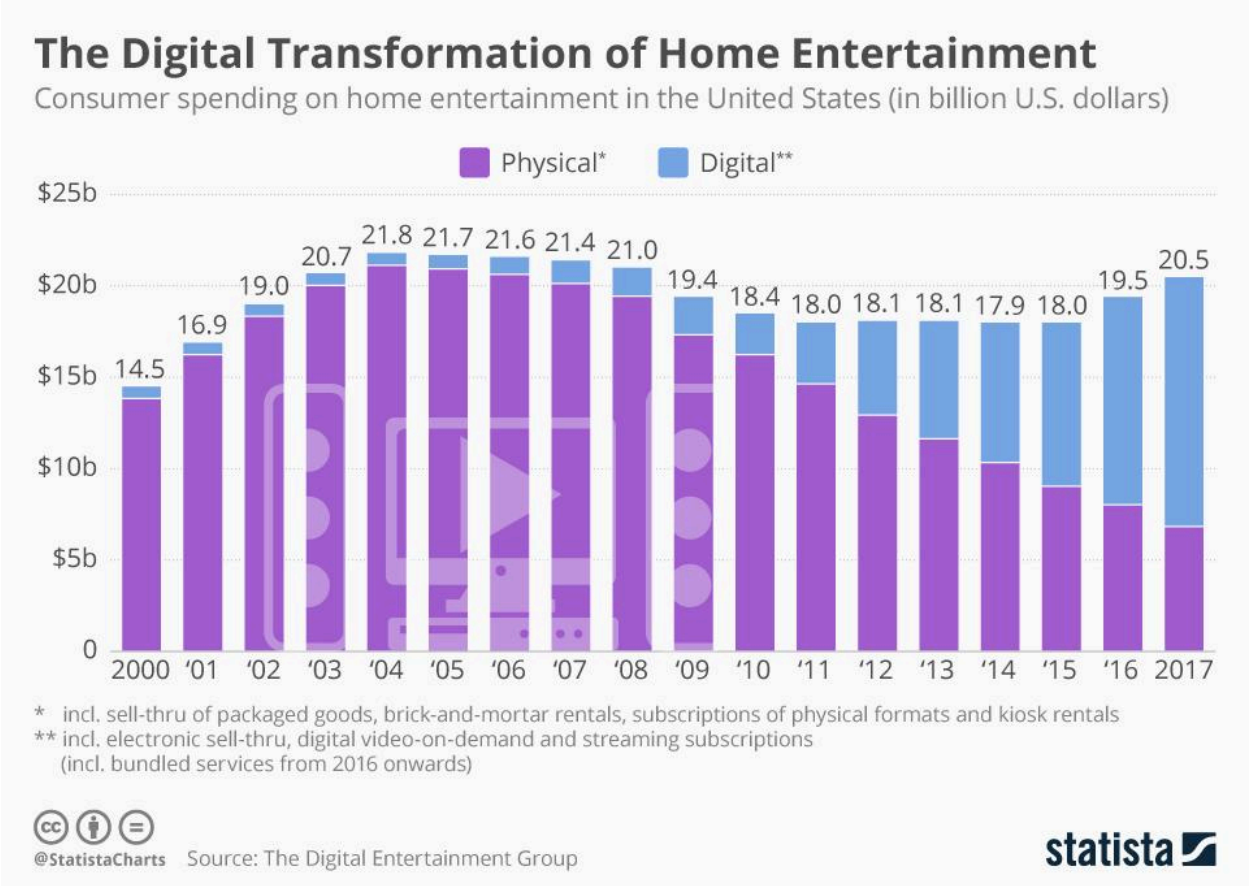

Figure 1: Growth in digital entertainment shows an increasing demand for novel entertainment

The idea of this application is to assist a user in playing a board or card game with physical pieces. Once our application is installed in a Nao robot, the robot is the Ro-board. The Ro-board serves as a second player for the game, so the user does not have to play by themselves. A personal assistant such as this could be used for solitary individuals or as

entertainment in caretaking. We will be using the development kit designed for these robots, utilizing both Python to create the game logic and C++ to control the Ro-board's movements. The application can be downloaded onto any Nao robot, once available on the Nao robot application store, to be used across the world. The Ro-board will have a few different functions to serve the player, such as playing the game, tutorial process, and interaction during the game. The Ro-board will have the capability to interact with the user through sound and gestures, as well as understanding the other player's words and actions. The Ro-board is not designed as a conversational robot and as such the interactions will be limited to calls and responses, prioritizing playing the game correctly and assisting the user in learning the game. We will start with basic functionality of the Ro-board, getting it to move from our programming and interact with the game board. Then, we will have the Ro-board interact with the game state and try to further its goal of completing the game. Once the Ro-board is able to play and finish a game, we can add more functions such as a tutorial and user interaction. The program structure for playing the game is designed as a state machine with weighted edges which will influence the Ro-board's decision on how to proceed. By changing these weights we can influence the logic of the Ro-board and make it easier or harder for the player.

Once a game has been started and the robot is ready to play, it will sit in idle waiting for the player(s) to make their move. Once they have completed their move, they can indicate this to the robot and the robot will take over. First, to get an overview of the state of the game, it will take a picture of the board and other game pieces. Then, it will analyze the image or images to determine what the state of the game is. For example, in Rummikub the robot would take an image of both its rack and the layout of the tiles and then process both of these images. In chess or checkers, this will just involve taking a single picture of the whole game board and then process the image. After determining the state of the game, if the game has completed, the robot will end its process. Otherwise, the robot will determine which move it will play. For example, in chess, there are well-defined AIs that can determine the best move (or subpar moves if the difficulty is lowered) from the game state. Then, it can perform this move and end its turn. If the robot's move ends the game, the robot's process will end. Otherwise, the robot will go idle again and wait for the player(s) to complete their turn (Figure 2).

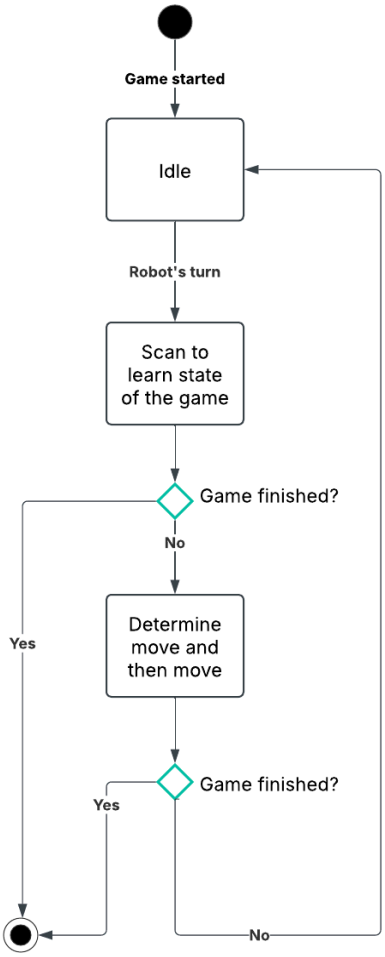

Figure 2: UML Activity Diagram displaying how the robot plays

One aspect of the project revolves around teaching the player how to play the game. If a player does not already know how to play the game, we want to implement a tutorial as part of our application that allows the player to learn the game from the Nao robot. The tutorial will begin with the robot explaining how to win the game. For example, in a game like Rummikub, the tutorial would explain and potentially demonstrate that the first player to get rid of all their tiles would win the game. Then, the robot would explain the setup, followed by the rules, and how to play the game. For a game like chess, this would include explaining the board, setting up the pieces in the correct spots, how the pieces move, legal and illegal moves in any position, and generally how to progress towards victory. After that, the robot will explain common strategies and potentially the best strategy for any player to win the game. In a card game like blackjack, this could include when and how to make certain actions like betting more, hitting, standing, etc. in addition to counting cards. Finally, the robot will ask the player if there are any questions about the game and repeat or re-explain anything that was previously explained or, if there are not any questions, allow the player to begin playing the game (Figure 3).

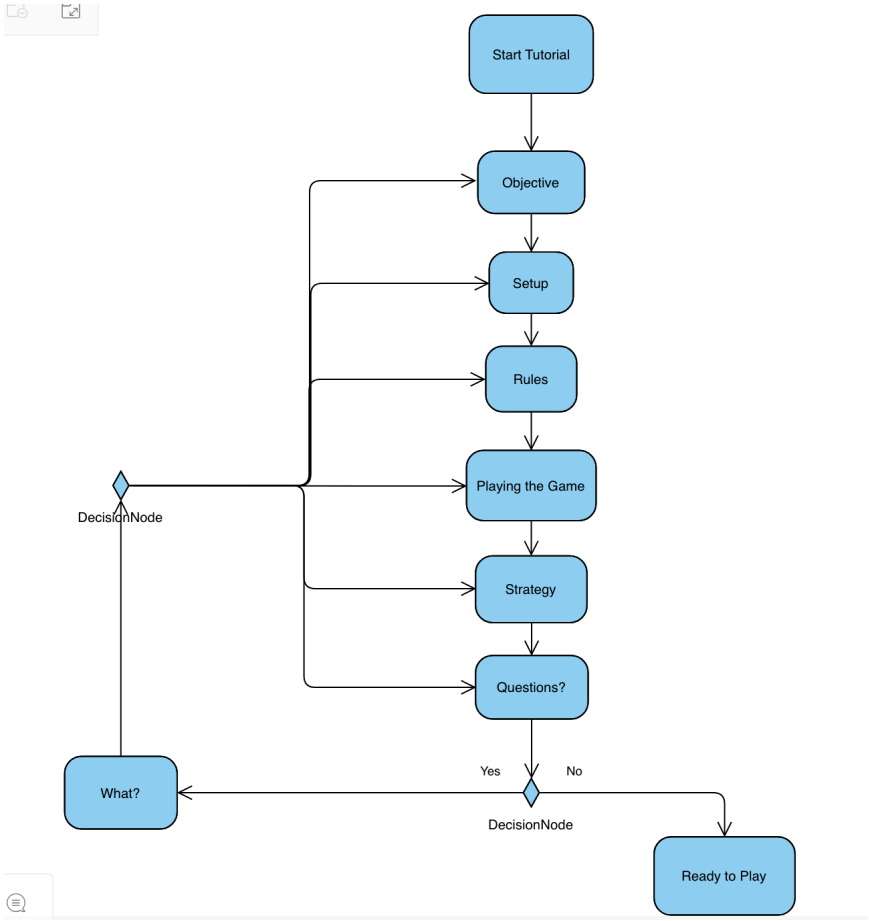

Figure 3: UML Activity Diagram displaying the robot's tutorial process

Another aspect that will be included in the robot is making the robot more “human-like”. At the basic functionality, the robot will just be idling and performing predictable actions that it only needs to play the game. This feature will add some much needed “life” to the robot. There are multiple ways we will accomplish this. One way is by using encouraging comments. For example, after a player makes their turn and it is the robot’s turn, the robot will sometimes tell the player that they made a good move. Or, at the end of the game, the robot can announce “Good game”. Furthermore, we can have the robot trash talk the opponent. When the player makes a move, the robot can tell them that they made a mistake. Or, after the robot makes a move, he can taunt the other player. Also, there could be an option for the robot to get mad when it loses, then throwing pieces or smashing the game board. Finally, the robot can also make “gestures”. For example, if it is making its move, it can move into a thinking pose. These actions serve to make the robot a better companion for the user and add some complexity to its actions.