**Development Process and Implementation**

Revision 1:

Author: Max Moore, Ernest Selman, Derek Savery, Andrew Danha, Eric Le Fort, Guy Meyer

Description: Creation of document and solidification of initial ideas

Date: Oct. 22/ 2016

Guy Meyer - 1320231 - meyerg

Ernest Selman - 1201291 - selmae

Eric Le Fort - 1308609 - leforte

Derek Savery - 1219142 - saverydj

Andrew Danha - 1223811 - danhaas

Maxwell Moore - 1320009 - moorem8

|  |  |  |  |
| --- | --- | --- | --- |
| Revision # | Author(s) | Description | Date |
| 1 | Max Moore  Ernest Selman  Derek Savery  Andrew Danha  Eric Le Fort  Guy Meyer | Creation of document and solidification  of initial ideas | Oct. 22/2016 |

·

Overall Process workflow

o What steps need to be done and in what order?

Our intention is for the hardware and software development to work on certain parts of the project at the same time.

|  |  |
| --- | --- |
| Software | Hardware |
| 1) Documentation   * Software Requirements Specification * High-Level Design * Low-Level Design   2) Simulation of Physical Grounds   * Perform tests for physical constraints * Design GUI for visualization * Implement basic physics engine * Test and set boundaries in safety   3) Visual Recognition (VR)   * Design physical system to take photos automatically * Design system to recognize position of balls relative to table edges * Be able to map the balls to the simulation to scale.   4) Artificial Intelligence (AI)   * Design a decision making strategy * Have the decision be mapped into the real world, with the correct parameters to be accepted by the corresponding hardware system ~ direction, power, position, etc.   5) User Guide  6) Advanced strategy   * Add to the Artificial intelligence system the ability to predict where the ball will land, and be able to take a shot placing the cue ball in a strategic position, such that either our opponent's next shot will be limited, or our own would be benefited. | 1. Create requirements for the mechanical portion of the system    * Collect measurements    * Address all safety requirements    * Address all functional requirements 2. Research prior creations of the project to develop ideas 3. Create CAD model of the workspace 4. Develop design alternatives in CAD using the workspace model for a given part of the robot 5. Select a design alternative based on feasibility, efficiency and availability of parts (actuators, motors, etc.) 6. Purchase necessary parts or create parts in custom design 7. Build physical implementation of the robot part 8. Begin interfacing the system component with the software system   (Note: Steps 4 to 8 are repeated for the next part of robot) |

o What inputs are needed for each step and what are the outputs for that step.

|  |  |
| --- | --- |
| Software | Hardware |
| 1. Inputs: None Outputs: Documentation and design 2. Inputs: physical pool table, cue, and balls for testing  Outputs: Software system to accurately simulate given shots based upon parameters such as power, direction, position, and our tables specific physical attributes. 3. Inputs: some sort of camera, and a pool table setup such that we can take reproducible photos from a static location. Outputs: a software system that maps the physical environment to a model simulated in code dynamically and automatically. 4. Inputs: a functional, shot taking robot, that can accept software inputs from our system.  Outputs: a system that can autonomously make reasonable decisions as to where to shoot, communicate that with the physical system, and take the shot somewhat accurately. 5. Inputs: Relevant completed user interfaces. Outputs: User guide. 6. Inputs: the completion and testing of an acceptable software system and Hardware interface, in short a complete project that we can show. Output: A smarter decision making Robot, that will be more difficult (and fun) to play against. | 1. Inputs: Physical pool table Outputs: Measurements and functional requirements 2. Inputs: Measurements and functional requirements Outputs: Possible design ideas 3. Inputs: Possible design ideas Outputs: CAD model of workspace 4. Inputs: CAD model of workspace Outputs: Multiple CAD design alternatives for given robot part 5. Inputs: Multiple CAD design alternatives for given robot part Outputs: One CAD design for robot part 6. Inputs: One CAD design for robot part Outputs: Physical components needed to build part 7. Inputs: Physical components needed to build part Outputs: To-scale implementation of part 8. Inputs: To-scale implementation of part Outputs: Functional part that is fully interfaced with the software system |

o What are the acceptance criteria for the each of the outputs of a step?

|  |  |
| --- | --- |
| Software | Hardware |
| 1)   * Documents are complete and agreed upon by the group and any advisors.   2)   * Our simulation passes mathematical testing and is displayed graphically.   3)   * The balls are correctly mapped in software automatically within 1mm accuracy of our actual measurements on a pre-set table. * Note: this should be tested in a physical system with whatever setup we intend to have the camera in.   4)   * Have a basic strategy implemented that, given the completion of the associated hardware, will be able to make a decision that will hit a desired ball 90% of the time, given a straight shot. 50% of the time these straight shots will be able to be sunk, in good conditions.   5)   * Giving a user the user guide is sufficient to inform the user on how to operate the system for 95% of users.   6)   * Have a strategy implemented that, given the completion of the associated hardware, will be objectively more difficult given the position of the cue ball 70% of the time after the robot is finished, and see some increase in the average number of balls the robots sinks in a row. | 1. Accurate measurements of the physical table, realistic functional requirements based on the physical attributes of the table 2. Realistic design ideas. Make sure the entire group is agreed on possible design ideas. 3. Representationally accurate model of the workspace 4. Realistic design alternatives. Multiple design alternatives. Make sure everyone in the group is agreed on design alternatives. 5. Have chosen one alternative which is decided to be the most feasible. 6. Have every component needed to build the part 7. Have finished part fully built 8. Have finished part fully interfaced with the software system |

· Provide some details on how each step should be done?

o What tools are you going to use? What versions?

o Are there any special instructions on settings or how to use the tools? What information to put in version control?

o Any standards you should follow? (e.g. coding standards)

|  |  |
| --- | --- |
| Software | Hardware |
| 1. LaTeX and document templates will be used to create the design documents. 2. The simulation of the real world will be done almost entirely within Java utilizing existing graphical libraries and physics libraries for realistic results. 3. VR will be accomplished using an Android app to gather visual data which will then be fed into a MATLAB VR library for computation. 4. Any optimisation done in the artificial intelligence will likely be done by MATLAB, but the bulk of it will be contained within Java, and the Java application which will have limited communication with the hardware. 5. LaTeX will be used to create necessary User documents. 6. This will require advanced optimisation algorithms which will be implemented as a complete subsystem in MATLAB. | 1. Tape measurer will be used for taking physical measurements. Metric system will be adhered to. 2. Google patents, project websites, youtube videos, past capstone projects 3. Autodesk Inventor Professional 2011 will be used 4. All CAD designs will be in scale with one another 5. Look for part specifications online, check feasibility of designs using dynamic simulations testing (i.e. simulink). 6. Parts will be ordered online and will be modified (if required) using machine shop tools 7. Finished part will be constructed using machine shop tools. 8. Standard communications protocol will be used for interfacing electro-mechanics with software |

**Version Control for Software Applications:**

The software Version control will be handled with Git Hub, where all of the code will be organised by the programming language that has been used for the portion, and subcategories based on modules that we think are applicable at the time.

Github will manage who added what to the code. When anything is added by anyone team member it should be tested independently by the other team members before continuation. Comments should be added to code if there is something that isn’t ready for testing or if there is something that needs particular attention when testing.

When testing is completed a note should be made which comments that a portion has been tested along with a date and version number. This should also be noted on Github. If something is tested and results are unsuccessful when compared with original expectations githubs “blame” function can be used. This “blame” function will be cleared once the problem has been resolved.

Comments should be kept on pieces of code where different testing iterations have taken place in the form:

“Date, test description,test ID(i.e. AI-4.1 for module 4 test one), success/failure, suggestions”

If large changes are made which are based upon this designation then it should be noted below the testing comments in the form:

“Date, change descriptions, because of which test”

If there have been changes they are definitely required to be tested, as such it will be an iterative process that ends in a set of successful tests.

Coding conventions will closely follow Google Style Guides for the appropriate languages.

**Version Control for Hardware Applications:**

The hardware version control will be handled in much the same way as the software (as mentioned above), by using GitHub to manage files, design files, CAD files, dynamics simulations files, etc. These will be organized based on their applicable subcategories. Github will manage who added what to the files throughout the lifetime of this project. Every file should be tested independently by the other members of the hardware team before continuing. Communication should be made to ensure that all files meet expectations and whether something needs further attention. If something is tested and seen to be unsuccessful against original expectations githubs “blame” function can be used, and cleared once the problem has been resolved.

Upon construction of the physical parts the version control functionality becomes a different issue since we are no longer operating exclusively in the digital realm. In dealing with physical parts it is likely that multiple members will be working on the part simultaneously. This aspect will likely help to ensure that satisfactory modifications to each part are being made. Furthermore, we will keep a log of who has performed what work so as to ensure that individuals may be held accountable for their actions. The log can be consulted so as to verify changes that have been made as well as classify change requests and bugs.