Autonomous Pool Playing Robot

Verification & Validation

Ernest Selman
selmae@mcmaster.ca
1201291

Eric Le Fort
leforte@mcmaster.ca
1308609

Guy Meyer
meyerg@mcmaster.ca
1320231

Andrew Danha
danhaas@mcmaster.ca
1223881

Max Moore
moorem8@mcmaster.ca
1320009

Derek Savery
saverydj@mcmaster.ca
1219142

March 3, 2017

Contents

1 Introduction **4**
1.1 Overview . . . 4
1.2 Purpose . . . 4
1.3 Naming Conventions & Definitions . . . 4
1.3.1 Definitions . . . 5
1.3.2 Acronyms & Abbreviations . . . 5

2 Traceability Matrix **6**

3 Mechanical Components **11**

4 Electrical System **17**

5 Software System **21**
5.1 Unit Tests . . . 21
5.1.1 PC Controller Program . . . 21
5.1.2 PC VR Program . . . 41
5.1.3 μC Program . . . 43
5.2 System Tests . . . 46

6 Summary of Results **50**

List of Tables

1 Revision History . . . 3
2 Definitions . . . 5
3 Acronyms and Abbreviations . . . 5
4 Functional Requirements Traceability Matrix - 1 . . . 6
5 Functional Requirements Traceability Matrix - 2 . . . 7
6 Non-Functional Requirements Traceability Matrix - 1 . . . 9
7 Non-Functional Requirements Traceability Matrix - 2 . . . 10
8 Synchronous Motion in X Rail . . . 11
9 Motion in Y Rail . . . 11
10 End-Effector Orientation . . . 12
11 Shooting Mechanism Orientation . . . 12
12 Perimeter Coverage . . . 12
13 Ball Avoidance . . . 13
14 Table Visibility . . . 13
15 System Obstruction . . . 13
16 System Weight . . . 14
17 Rigidity of Machine Body . . . 14
18 Transformer Stability . . . 14
19 User Proximity Safety . . . 15
20 Shut Down Buttons . . . 15
21 Striking Force - Strong . . . 15
22 Striking Force - Soft . . . 16
23 Sufficient Acceleration and Stepping Consistency . . . 16

24 User Input to Arduino . . . 17
25 Current Physical State: X-Rail . . . 17
26 Current Physical State: Y-Rail . . . 18
27 Current Physical State: Rotation . . . 18
28 Current Physical State: End-Effector . . . 18
29 Check for Exposed Circuitry . . . 19
30 Sensitive Component Isolation from High Voltage . . . 19
31 Voltage Regulation . . . 19
32 Circuit Breakers . . . 20
33 AC/DC Converter . . . 20
34 Power Supply from Standard Socket . . . 20
35 Ball Constructor Good Inputs . . . 21
36 Ball Constructor Large X . . . 22
37 Ball Constructor Large Y . . . 22
38 Ball Constructor Small X . . . 23
39 Ball Constructor Small Y . . . 23
40 Ball Constructor Small Value . . . 24
41 Ball Constructor Large Value . . . 24
42 Updating Table State . . . 25
43 Selecting an Optimal Shot . . . 25
44 Read Valid Table State from File . . . 26
45 Read Table State from Non-Existent File . . . 26
46 Read Table State from File with Invalid Data . . . 27
47 Initiating the VR Program . . . 27
48 Shot Constructor Good Inputs . . . 28
49 Shot Constructor Large X . . . 28
50 Shot Constructor Small X . . . 29
51 Shot Constructor Large Y . . . 29
52 Shot Constructor Small Y . . . 30
53 Shot Constructor Large Angle . . . 30
54 Shot Constructor Small Angle . . . 31
55 Shot Constructor Large Power . . . 31
56 Shot Constructor Small Power . . . 32
57 Simulation Instance Constructor Good Inputs . . . 32
58 Simulation Instance Constructor Good Inputs . . . 33
59 Simulation Instance Constructor Large Power . . . 33
60 Check for Walls . . . 34
61 Get Angle from Coordinates . . . 35
62 Ball-Wall Collision . . . 36
63 Check if in Pocket . . . 37
64 TableState Constructor Good Inputs . . . 38
65 TableState Constructor Too Many Elements . . . 38
66 TableState Constructor Not Enough Elements . . . 39
67 TableState Constructor Elements Too Small . . . 39
68 TableState Constructor Elements Too Large . . . 40
69 TableState Deep Copy . . . 40
70 Test Title . . . 41
71 Test Title . . . 42
72 Signal steps for X Motion . . . 43
73 Signal steps for Y Motion . . . 43
74 Signal steps for Rotational Motion . . . 44
75 Calculation of Steps Required . . . 44
76 Signal for Pneumatic Extension . . . 44
77 Signal for Pneumatic Extension . . . 45

78 Signal Steps for Rotational Motion of Air Flow Controller . . . 45
79 Aligned Shot . . . 46
80 Angled Shot . . . 46
81 Shot Cancelled Before Motion . . . 47
82 Shot Cancelled During Motion . . . 47
83 Move Request (To Zero X-Coordinate) . . . 48
84 Move Request (To Largest X-Coordinate) . . . 48
85 Shot Power Modification . . . 49
86 Check For Legality and Political Correctness . . . 49
87 Assessment of Durability . . . 50

Date	Revision #	Comments	Authors
27/02/2017	0	- Initial document creation	Eric Le Fort

Table 1: Revision History

1 Introduction

This document will provide a specification of a test plan for an automated pool playing robot and report on the results of that plan.

1.1 Overview

This document breaks down the required testing for each domain of the system. It begins with the hardware aspect, then moves to the electrical side and then finishes with software. Each section will go into further detail to describe each test case. Lastly, a summary of the results of testing will be provided to conclude the document.

1.2 Purpose

The aim of this document is to illuminate any design flaws, software bugs, or other issues in the system. Once these issues are discovered, the engineering team will be able to work on eliminating them or minimizing their frequency and consequences.

1.3 Naming Conventions & Definitions

This section outlines the various definitions, acronyms and abbreviations that will be used throughout this document in order to familiarize the reader prior to reading.

1.3.1 Definitions

Table 2 lists the definitions used in this document. The definitions given below are specific to this document and may not be identical to definitions of these terms in common use. The purpose of this section is to assist the user in understanding the requirements for the system.

Term	**Meaning**
X-axis	Distance along the length of the pool table
Y-axis	Distance across the width of the pool table
Z-axis	Height above the pool table
End-effector	The end of the arm that will strike the cue ball
θ	Rotational angle of end-effector
Cue	End-effector
Personal Computer	A laptop that will be used to run the more involved computational tasks such as visual recognition and the shot selection algorithm
Camera	Some form of image capture device (e.g. a digital camera, smartphone with a camera, etc.)
Table State	The current positions of all the balls on the table
Entity	Classes that have a state, behaviour and identity (e.g. Book, Car, Person, etc.)
Boundary	Classes that interact with users or external systems
Double	Double-precision floating point numbers

Table 2: Definitions

1.3.2 Acronyms & Abbreviations

Table 3 lists the acronyms and abbreviations used in this document.

Acronym/Abbreviation	**Meaning**
VR	Visual Recognition
PC	Personal Computer
μC	Micro-Controller
CRC	Class Responsibility Collaboration
TBT	To Be Tested

Table 3: Acronyms and Abbreviations

2 Traceability Matrix

The following traceability matrices will demonstrate that the tests to be performed prove that each of the specified requirements have been tested.

Functional Requirements Traceability Matrix

Req IDs	Reqs Tested	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18
Test Cases	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Tested Implicitly																			
1.1													X			X	X		
1.2													X			X	X		
1.3													X				X		
1.4																			
1.5																			
1.6																			
1.7																			
1.8																			
1.9																			
1.10																			
1.11																			
1.12																			
1.13																			
1.14																			
1.15																			
1.16																			
1.17																			
2.1																			
2.2																			
2.3																			
2.4																			
2.5																			
2.6																			
2.7																			
2.8																			
2.9																			
2.10																			
2.11																			
3.1.1																			
3.1.2																			
3.1.3																			
3.1.4																			
3.1.5																			
3.1.6																			
3.1.7																			
3.1.8																			

Table 4: Functional Requirements Traceability Matrix - 1

Req IDs	Reqs Tested	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18
3.1.9																			
3.1.10																			
3.1.11																			
3.1.12																			
3.1.13																			
3.1.14																			
3.1.15																			
3.1.16																			
3.1.17																			
3.1.18																			
3.1.19																			
3.1.20																			
3.1.21																			
3.1.22																			
3.1.23																			
3.1.24																			
3.1.25																			
3.1.26																			
3.1.27																			
3.1.28																			
3.1.29																			
3.1.30																			
3.1.31																			
3.1.32																			
3.1.33																			
3.1.34																			
3.1.35																			
3.2.1																			
3.2.2																			
3.3.1																			
3.3.2																			
3.3.3																			
3.3.4																			
3.3.5																			
3.3.6																			
3.3.7																			
4.1																			
4.2																			
4.3																			
4.4																			
4.5																			
4.6																			
4.7																			
4.8																			
4.9																			

Table 5: Functional Requirements Traceability Matrix - 2

Non-Functional Requirements Traceability Matrix

Req IDs	Reqs Tested	LF1	UH1	UH2	UH3	P1	P2	P3	P4	OE1	MS1	MS2	S1	S2	S3	S4	S5	S6	S7	CP1	L1
Test Cases	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Tested Implicitly																					
1.1			X			X															
1.2			X			X															
1.3			X																		
1.4																					
1.5																					
1.6																					
1.7																					
1.8																					
1.9																					
1.10																					
1.11																					
1.12																					
1.13																					
1.14																					
1.15																					
1.16																					
1.17																					
2.1																					
2.2																					
2.3																					
2.4																					
2.5																					
2.6																					
2.7																					
2.8																					
2.9																					
2.10																					
2.11																					
3.1.1																					
3.1.2																					
3.1.3																					
3.1.4																					
3.1.5																					
3.1.6																					
3.1.7																					
3.1.8																					
3.1.9																					
3.1.10																					
3.1.11																					
3.1.12																					
3.1.13																					

Table 6: Non-Functional Requirements Traceability Matrix - 1

Req IDs	Reqs Tested	LF1	UH1	UH2	UH3	P1	P2	P3	P4	OE1	MS1	MS2	S1	S2	S3	S4	S5	S6	S7	CP1	L1
3.1.14																					
3.1.15																					
3.1.16																					
3.1.17																					
3.1.18																					
3.1.19																					
3.1.20																					
3.1.21																					
3.1.22																					
3.1.23																					
3.1.24																					
3.1.25																					
3.1.26																					
3.1.27																					
3.1.28																					
3.1.29																					
3.1.30																					
3.1.31																					
3.1.32																					
3.1.33																					
3.1.34																					
3.1.35																					
3.2.1																					
3.2.2																					
3.3.1																					
3.3.2																					
3.3.3																					
3.3.4																					
3.3.5																					
3.3.6																					
3.3.7																					
4.1																					
4.2																					
4.3																					
4.4																					
4.5																					
4.6																					
4.7																					
4.8																					
4.9																					

Table 7: Non-Functional Requirements Traceability Matrix - 2

3 Mechanical Components

Test ID: 1.1	**Synchronous Motion in X Rail**	Status: TBT
Description: Verify that X-Rails can synchronously move to the same location at the same speed without getting stuck while loaded		
Pass/Fail Condition: If rail moves adequately and quickly as expected		
Pre-Conditions: None Input: Location along x-direction (i.e. 2000 steps)		
Expected Results: Smooth and consistent motion along axis until position is met. Followed by an immediate stop	Actual Results:	
Post-Conditions: Rails are stationary with no slip.		

Table 8: Synchronous Motion in X Rail

Test ID: 1.2	**Motion in Y Rail**	Status: TBT
Description:Verify that Y-Rail can move to a location without getting stuck while loaded		
Pass/Fail Condition: If rail moves adequately and quickly as expected		
Pre-Conditions: None Input: Location along y-direction		
Expected Results: Smooth and consistent motion along axis until position is met. Followed by an immediate stop	Actual Results:	
Post-Conditions: Rail is stationary with no slip.		

Table 9: Motion in Y Rail

Test ID: 1.3	**End-Effector Orientation**	Status: TBT
Description: Verify that EE-Base Motor can orient to a specific angle without getting stuck while loaded		
Pass/Fail Condition: If motor turns adequately and quickly as expected to correct angle		
Pre-Conditions: None Input: Angle of orientation with respect to the x-axis		
Expected Results: Smooth and consistent motion until orientation is met. Followed by an immediate stop	Actual Results:	
Post-Conditions: Motor is stationary.		

Table 10: End-Effector Orientation

Test ID: 1.4	**Shooting Mechanism Orientation**	Status: TBT
Description: EE is positioned correctly and waiting command to power piston		
Pass/Fail Condition: Piston is settled at correct oreintation, awaiting command to actuate piston		
Pre-Conditions: Motors orient piston to proper orentation Input: Position and orientation components sent to Arduino		
Expected Results: System moves to desired location and waits for piston signal	Actual Results:	
Post-Conditions: Piston can be safely actauted and strike cue ball		

Table 11: Shooting Mechanism Orientation

Test ID: 1.5	**Perimeter Coverage**	Status: TBT
Description: EE will be moved around the table to ensure that it is able to reach all locations and orientations		
Pass/Fail Condition: EE is capable of completing a full trip around the perimeter without stops		
Pre-Conditions: None Input: Motion command from Arduino		
Expected Results: EE will travel around perimeter of table. Inspection that its location is sufficient for shot-taking is required.	Actual Results:	
Post-Conditions: System awaits next command.		

Table 12: Perimeter Coverage

Test ID: 1.6	**Ball Avoidance**	Status: TBT
Description: As the EE is moving around the table it much avoid the balls to not interfere with gameplay		
Pass/Fail Condition: Able to move randomly around table without moving rolling or stationary balls		
Pre-Conditions: Ball in motion OR stationary Input: Random motion along table		
Expected Results: EE travels directly over balls and does not make contact	Actual Results:	
Post-Conditions: None		

Table 13: Ball Avoidance

Test ID: 1.7	**Table Visibility**	Status: TBT
Description: The amount of table visible is approximated.		
Pass/Fail Condition: This test is passed if players are able to see 100% table setup upon their turn.		
Pre-Conditions: Machine is in a position where it is ready for a "Take a Shot" command. Input: Percentage visibility of the table.		
Expected Results: Player can see 100% of the table without excessive effort or movement.	Actual Results:	
Post-Conditions: None.		

Table 14: Table Visibility

Test ID: 1.8	**System Obstruction**	Status: TBT
Description: The machine will be placed in positions which make it as difficult as possible to take a shot. The difficulty of the shot will then be determined.		
Pass/Fail Condition: This test is passed if the design of the machine allows users to take any shot they would normally be able to make.		
Pre-Conditions: The machine and balls should be setup in a way that makes a shot as difficult as possible. Input: Difficulty of shot.		
Expected Results: Player is able to make their shot with no more than a low degree of difficulty relative to the shot difficulty without the machine.	Actual Results:	
Post-Conditions: None.		

Table 15: System Obstruction

Test ID: 1.9	**System Weight**	Status: TBT
Description: The components of the machine will be weighed and those weights will be added together to get the total weight.		
Pass/Fail Condition: This test is passed if the weight of the machine is less than 250 lbs.		
Pre-Conditions: None. Input: Weights of all components used.		
Expected Results: Machine weighs less than 250 lbs.	Actual Results:	
Post-Conditions: None.		

Table 16: System Weight

Test ID: 1.10	**Rigidity of Machine Body**	Status: TBT
Description: The machine must be rigid such that nominal strain < 0.1		
Pass/Fail Condition: This test is passed if the body of the machine is rigid such that nominal strain < 0.1		
Pre-Conditions: None. Input: The impulse from the strongest shot on the machine in multiple locations and directions.		
Expected Results: The machine body should not suffer deformation greater than magnitude 0.1 nominal strain.	Actual Results:	
Post-Conditions: The machine body should return to its initial state.		

Table 17: Rigidity of Machine Body

Test ID: 1.11	**Transformer Stability**	Status: TBT
Description: Machine will move around the table as sharply as possible in typical execution and the transformer will be checked for stability.		
Pass/Fail Condition: This test is passed if the transformer remains sturdy and secured.		
Pre-Conditions: None. Input: Quickest movement along the table in each direction.		
Expected Results: The transformer remains secured in position.	Actual Results:	
Post-Conditions: None.		

Table 18: Transformer Stability

Test ID: 1.12	**User Proximity Safety**	Status: TBT
Description: The machine will move to the furthest points it can reach and the distance from the table will be measured.		
Pass/Fail Condition: This test is passed if the machine is never further than 2 ft away from the table.		
Pre-Conditions: None. Input: End-effector moved in various locations to test the extreme distances it can reach.		
Expected Results: Mechanism extends less than 2ft from the perimeter of the table at all times.	Actual Results:	
Post-Conditions: None.		

Table 19: User Proximity Safety

Test ID: 1.13	**Shut Down Button Locations**	Status: TBT
Description: The distance from pinch points to a stop button is measured.		
Pass/Fail Condition: This test is passed if there are shut down buttons located within the smallest reach of a typical adult of pinch points.		
Pre-Conditions: None. Input: The distance from pinch points when the system is moved to various positions.		
Expected Results: Shut down buttons are always less than the smallest reach of a typical adult from pinch points.	Actual Results:	
Post-Conditions: None.		

Table 20: Shut Down Buttons

Test ID: 1.14	**Striking Force - Strong**	Status: TBT
Description: Ensure shot is strong enough so that the cue ball can reach the whole table with sufficient force		
Pass/Fail Condition: At maximum strength the cue ball can cover the length of the table and return to half after hitting a bank		
Pre-Conditions: Cue ball placed along one maximum x position Input: Maximum strength shot		
Expected Results: Cue ball rolls across long edge of table, reflects of the bank and returns to half	Actual Results:	
Post-Conditions: Balls are stationary and Shooting mechanism is retracted		

Table 21: Striking Force - Strong

Test ID: 1.15	**Striking Force - Soft**	Status: TBT
Description: Ensure shot is soft enough so that the cue ball can reach nearby balls with control		
Pass/Fail Condition: At minimum/low strength the cue ball can lightly strike a nearby ball (within 20 cm) while moving no more than 20 cm after the hit		
Pre-Conditions: Cue ball placed within 20 cm of another ball Input: Minimum stength shot		
Expected Results: Cue ball rolls towards other ball, makes contact and quickly comes to a stop	Actual Results:	
Post-Conditions: Balls are stationary and Shooting mechanism is retracted		

Table 22: Striking Force - Soft

Test ID: 1.16	**Sufficient Acceleration and Stepping Consistency**	Status: TBT
Description: At maximum loading capacity the system can accelerate to a terminal speed at which the EE is moved quickly enough		
Pass/Fail Condition: While the physcial construction is finished the system will be told to move long distances several times to ensure repeatability and consistency in acceleration		
Pre-Conditions: System is stationary Input: Move EE between opposite corners multiple times (x10 cycles)		
Expected Results: After completion the EE should return to its original location within a couple of steps	Actual Results:	
Post-Conditions: Balls are stationary and Shooting mechanism is retracted		

Table 23: Sufficient Acceleration and Stepping Consistency

4 Electrical System

Test ID: 2.1	**User Input to Arduino**	Status: TBT
Description: User applies input, then the Arduino indicates a message was received		
Pass/Fail Condition: Arduino output to console correct desired status		
Pre-Conditions: None Input: User pressed input button		
Expected Results: Related console output: make shot, cancel, or move, depending on the button pressed	Actual Results:	
Post-Conditions: None		

Table 24: User Input to Arduino

Test ID: 2.2	**Current Physical State: X-Rail**	Status: TBT
Description: Verify that the system can detect the machine's current physical state at certain locations along the x-rail.		
Pass/Fail Condition: This condition is passed if both sensors are triggered.		
Pre-Conditions: None Input: Attempt to move system along the x-rail to the lower-limit position then the upper limit position.		
Expected Results: X-rail sensors indicate that the system is in lower-limit/upper-limit positions and motion is stopped.	Actual Results:	
Post-Conditions: None		

Table 25: Current Physical State: X-Rail

Test ID: 2.3	**Current Physical State: Y-Rail**	Status: TBT
Description: Verify that the system can detect the machine's current physical state at certain locations along the y-rail.		
Pass/Fail Condition: This condition is passed if both sensors are triggered.		
Pre-Conditions: None Input: Attempt to move system along the y-rail to the lower-limit position then the upper limit position.		
Expected Results: Y-rail sensors indicate that the system is in lower-limit/upper-limit positions and motion is stopped.	Actual Results:	
Post-Conditions: None		

Table 26: Current Physical State: Y-Rail

Test ID: 2.4	**Current Physical State: Rotation**	Status: TBT
Description: Verify that the system can detect the machine's current physical state at certain angular positions.		
Pass/Fail Condition: This condition is passed if the sensor indicates that the system in the position the machine is actually in to within 0.3 degrees.		
Pre-Conditions: None Input: Rotate the end-effector to various set positions.		
Expected Results: Sensor indicates that the system is in reference position.	Actual Results:	
Post-Conditions: None		

Table 27: Current Physical State: Rotation

Test ID: 2.5	**Current Physical State: End-Effector**	Status: TBT
Description: Verify that the system can detect the machine's current physical state at certain locations along the end-effector's range of motion.		
Pass/Fail Condition: This condition is passed if the sensors indicate that the system in in the target position within 2 millimetres.		
Pre-Conditions: None Input: Predetermined target locations		
Expected Results: End-effector sensors indicate that the end-effector is in the target location.	Actual Results:	
Post-Conditions: None		

Table 28: Current Physical State: End-Effector

Test ID: 2.6	**Check for Exposed Circuitry**	Status: TBT
Description: Circuitry will be inspected to ensure none is exposed.		
Pass/Fail Condition: This test is passed if no circuitry is exposed.		
Pre-Conditions: None. Input: Result of wire inspection.		
Expected Results: No exposed circuitry.	Actual Results:	
Post-Conditions: None.		

Table 29: Check for Exposed Circuitry

Test ID: 2.7	**Sensitive Component Isolation from High Voltage**	Status: TBT
Description: The voltage near sensitive components will be measured to ensure they are at safe levels.		
Pass/Fail Condition: This test is passed if wires connected to sensitive components fall within their maximum parameters as specified by the device.		
Pre-Conditions: None. Input: Inspect wires connected to electrical equipment stated above.		
Expected Results: All components are isolated from unsafely high voltage.	Actual Results:	
Post-Conditions: None.		

Table 30: Sensitive Component Isolation from High Voltage

Test ID: 2.8	**Voltage Regulation**	Status: TBT
Description: The circuit to the μC will be provided various voltages and t. Pass/Fail Condition: This test is passed if the output voltage from the transformer is within the required μC voltage requirements.		
Pre-Conditions: None. Input: Reading of voltage fed into μC using a multimeter.		
Expected Results: Voltage is within 7 V DC - 12 V DC.	Actual Results:	
Post-Conditions: None.		

Table 31: Voltage Regulation

Test ID: 2.9	**Circuit Breakers**	Status: TBT
Description: High voltage will be applied to components to ensure that the circuit breakers perform as expected.		
Pass/Fail Condition: This test is passed if the circuits to all high voltage components are broken before unsafe voltage is applied.		
Pre-Conditions: None. Input: Sufficiently hight voltage.		
Expected Results: All circuits with unsafe voltages are broken.	Actual Results:	
Post-Conditions: None.		

Table 32: Circuit Breakers

Test ID: 2.10	**AC/DC Converter**	Status: TBT
Description: Verify that the transformer converts 110 AC, 60 Hz to DC ranges that power the μC at the appropriate voltage.		
Pass/Fail Condition: This condition is passed if the output voltage is a DC voltage within 7-12		
Pre-Conditions: None Input: Multimeter output voltage readings from the transformer.		
Expected Results: The output voltage is a DC voltage within 7 - 12 VDC	Actual Results:	
Post-Conditions: None		

Table 33: AC/DC Converter

Test ID: 2.11	**Power Supply from Standard Socket**	Status: TBT
Description: The system will be plugged into a standard wall socket and functionality will be assessed.		
Pass/Fail Condition: All components of the system are supplied with sufficient power.		
Pre-Conditions: None Input: The power from a standard wall socket.		
Expected Results: The system has enough power to perform normally.	Actual Results:	
Post-Conditions: None		

Table 34: Power Supply from Standard Socket

5 Software System

The software system is comprised of four main components: a control system running on an Arduino microcontroller, an automated image capture application running on an Android smartphone, as well a visual recognition program and smart shot selection program running on a PC. On top of the typical suite of unit tests to verify correctness of methods, rigorous system testing will also be crucial to adequately test this system.

5.1 Unit Tests

This section will provide a plethora of test cases which aim to prove correctness of the program. Each individual class will be tested in order to make finding specific test cases easier.

5.1.1 PC Controller Program

Ball Tests

Test ID: 3.1.1	Module: Ball	Status: TBT
Ball Constructor Good Inputs		
Description: Builds a new Ball object.		
Pass/Fail Conditions: This test is passed if all the fields inside of Ball are correctly initialized.		
Pre-Conditions: None Input: 1, 0.7, 0		
Expected Results: A new ball with x-coordinate 1, y-coordinate 0.7, and the value 0.	Actual Results:	
Post-Conditions: A new Ball object should be available.		

Table 35: Ball Constructor Good Inputs

Test ID: 3.1.2	Module: Ball	Status: TBT
Ball Constructor Large X		
Description: Builds a new Ball object with an x-coordinate that is too large.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown		
Pre-Conditions: None Input: 1.87658, 0.7, 0		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 36: Ball Constructor Large X

Test ID: 3.1.3	Module: Ball	Status: TBT
Ball Constructor Large Y		
Description: Builds a new Ball object with a y-coordinate that is too large.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown		
Pre-Conditions: None Input: 1, 0.94958, 0		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 37: Ball Constructor Large Y

Test ID: 3.1.4	Module: Ball	Status: TBT
Ball Constructor Small X		
Description: Builds a new Ball object with an x-coordinate that is too small.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown		
Pre-Conditions: None Input: -1.001, 0.7, 0		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 38: Ball Constructor Small X

Test ID: 3.1.5	Module: Ball	Status: TBT
Ball Constructor Small Y		
Description: Builds a new Ball object with a y-coordinate that is too small.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown		
Pre-Conditions: None Input: 1, -1.001, 0		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 39: Ball Constructor Small Y

Test ID: 3.1.6	Module: Ball	Status: TBT
Ball Constructor Small Value		
Description: Builds a new Ball object with a value that is too small.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown		
Pre-Conditions: None Input: 1, 0.7, -1		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 40: Ball Constructor Small Value

Test ID: 3.1.7	Module: Ball	Status: TBT
Ball Constructor Large Value		
Description: Builds a new Ball object with a value that is too large.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None Input: 1, 0.7, 16		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 41: Ball Constructor Large Value

InferenceEngine Tests

Test ID: 3.1.8	Module: InferenceEngine	Status: TBT
Updating Table State		
Description: Updates the current table state being tested.		
Pass/Fail Conditions: This test is passed if all post-conditions are met.		
Pre-Conditions: None Input: A 16-by-2 array of doubles that are valid positions, BallType.STRIPES		
Expected Results: None	Actual Results: None	
Post-Conditions: 1. Stored BallType is BallType.STRIPES. 2. The stored positions array is the same as the one passed in. 3. The stored best shot is null. 4. The stored table state reflects the positions passed in.		

Table 42: Updating Table State

Test ID: 3.1.9	Module: InferenceEngine	Status: TBT
Selecting an Optimal Shot		
Description: Runs the method which simulates all direct shots that can be made.		
Pass/Fail Conditions: This test is passed if a reasonable Shot is returned.		
Pre-Conditions: The current table state is not null and the current ball type is not null or BallType.CUE. Input: None		
Expected Results: A reasonable Shot (no bank shots, shooting the right ball, valid x-/y-coordinates).	Actual Results:	
Post-Conditions: The best shot for the current table state is stored.		

Table 43: Selecting an Optimal Shot

PCCommunicator Tests

Test ID: 3.1.10	Module: PCCommunicator	Status: TBT
Read Valid Table State from File		
Description: Reads a table state from a file.		
Pass/Fail Conditions: This test is passed if the output matches the data in the text file.		
Pre-Conditions: None. Input: A text file with 16 ball positions		
Expected Results: The 16 ball positions stored in the text file.	Actual Results:	
Post-Conditions: None.		

Table 44: Read Valid Table State from File

Test ID: 3.1.11	Module: PCCommunicator	Status: TBT
Read Table State from Non-Existent File		
Description: Attempts to read from a non-existent table state file.		
Pass/Fail Conditions: This test is passed if a FileNotFoundException is thrown.		
Pre-Conditions: None. Input: None.		
Expected Results: A FileNotFoundException is thrown.	Actual Results:	
Post-Conditions: None.		

Table 45: Read Table State from Non-Existent File

Test ID: 3.1.12	Module: PCCommunicator	Status: TBT
Read Table State from File with Invalid Data		
Description: Attempts to read from a file that is not correctly formatted.		
Pass/Fail Conditions: This test is passed if an InputMismatchException is thrown.		
Pre-Conditions: None. Input: A file containing the text “Bad data”.		
Expected Results: An InputMismatchException is thrown.	Actual Results:	
Post-Conditions: None.		

Table 46: Read Table State from File with Invalid Data

Test ID: 3.1.13	Module: PCCommunicator	Status: TBT
Initiating the VR Program		
Description: Runs the method which automatically invokes the VR program.		
Pass/Fail Conditions: The test is passed if the VR Program has been run.		
Pre-Conditions: None. Input: None.		
Expected Results: Program is run and TableState.csv has been updated.	Actual Results:	
Post-Conditions: TableState.csv contains the results of the VR Program.		

Table 47: Initiating the VR Program

Shot Tests

Test ID: 3.1.14	Module: Shot	Status: TBT
Shot Constructor Good Inputs		
Description: Builds a new Shot.		
Pass/Fail Conditions: This test is passed if the Shot is successfully created and stores the correct information.		
Pre-Conditions: None Input: 1, 0.5, 3.5, 1		
Expected Results: A new Shot with an x-coordinate of 1, a y-coordinate of 0.5, an angle of 3.5, and a power of 1.	Actual Results:	
Post-Conditions: Shot has been created.		

Table 48: Shot Constructor Good Inputs

Test ID: 3.1.15	Module: Shot	Status: TBT
Shot Constructor Large X		
Description: Builds a new Shot with an x-value that is too large.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None Input: 1.87658, 0.5, 3.5, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 49: Shot Constructor Large X

Test ID: 3.1.16	Module: Shot	Status: TBT
Shot Constructor Small X		
Description: Builds a new Shot with an x-value that is too small.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None Input: -0.001, 0.5, 3.5, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 50: Shot Constructor Small X

Test ID: 3.1.17	Module: Shot	Status: TBT
Shot Constructor Large Y		
Description: Builds a new Shot with a y-value that is too large.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None Input: 1, 0.94958, 3.5, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 51: Shot Constructor Large Y

Test ID: 3.1.18	Module: Shot	Status: TBT
Shot Constructor Small Y		
Description: Builds a new Shot with a y-value that is too small.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None Input: 1, -0.001, 3.5, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 52: Shot Constructor Small Y

Test ID: 3.1.19	Module: Shot	Status: TBT
Shot Constructor Large Angle		
Description: Builds a new Shot with an angle that is too large.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None Input: 1, 0.5, 6.284, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 53: Shot Constructor Large Angle

Test ID: 3.1.20	Module: Shot	Status: TBT
Shot Constructor Small Y		
Description: Builds a new Shot with an angle that is too small.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None Input: 1, 0.5, -0.01, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 54: Shot Constructor Small Angle

Test ID: 3.1.21	Module: Shot	Status: TBT
Shot Constructor Large Power		
Description: Builds a new Shot with a power that is too large.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None Input: 1, 0.5, 3.5, 1.001		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 55: Shot Constructor Large Power

Test ID: 3.1.22	Module: Shot	Status: TBT
Shot Constructor Small Power		
Description: Builds a new Shot with a power that is too small.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None Input: 1, 0.5, 3.5, 0		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 56: Shot Constructor Small Power

SimulationInstance Tests

Test ID: 3.1.23	Module: SimulationInstance	Status: TBT
Simulation Instance Constructor Good Inputs Not Shooting 8-Ball		
Description: Builds a new SimulationInstance that is not shooting for the 8-ball.		
Pass/Fail Conditions: This test is passed if the array of Balls is created, the 8-ball is not the target ball, and the initial velocity of the cue ball is set.		
Pre-Conditions: InferenceEngine.myBallType = BallType.SOLID Input: A 16-by-2 array of doubles with at least one ball of type "solid" on the table, 2, 0.4		
Expected Results: A SimulationInstance has been created with an array of Balls with positions corresponding to the array, the initial velocity vectors of the cue ball have been set according to the power and angle.	Actual Results:	
Post-Conditions: A SimulationInstance has been created.		

Table 57: Simulation Instance Constructor Good Inputs

Test ID: 3.1.24	Module: SimulationInstance	Status: TBT
Simulation Instance Constructor Good Inputs Shooting 8-Ball		
Description: Builds a new SimulationInstance that is shooting for the 8-ball.		
Pass/Fail Conditions: This test is passed if the array of Balls is created, the 8-ball is the target ball, and the initial velocity of the cue ball is set.		
Pre-Conditions: InferenceEngine.myBallType = BallType.SOLID Input: A 16-by-2 array of doubles with no balls of type "solid" on the table, 2, 0.4		
Expected Results: A SimulationInstance has been created with an array of Balls with positions corresponding to the array, the initial velocity vectors of the cue ball have been set according to the power and angle.	Actual Results:	
Post-Conditions: A SimulationInstance has been created.		

Table 58: Simulation Instance Constructor Good Inputs

Test ID: 3.1.25	Module: SimulationInstance	Status: TBT
Simulation Instance Constructor Large Power		
Description: Builds a new SimulationInstance with a power that is too large.		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException has been thrown.		
Pre-Conditions: None Input: A 16-by-2 array of doubles, 2, 1.001		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: An IllegalArgumentException has been thrown.		

Table 59: Simulation Instance Constructor Large Power

Test ID: 3.1.26	Module: SimulationInstance	Status: TBT

Check for Walls

Description: Runs the method which checks for a wall at the given coordinates.

Pass/Fail Conditions: This test is passed if the expected results are equal to the actual results.

Pre-Conditions: None

Inputs:
(0.07070, true)
(0.07072, true)
(0.866, true)
(0.868, true)
(0.980, true)
(0.982, true)
(1.776, true)
(1.778, true)
(0.07070, false)
(0.07072, false)
(0.849, false)
(0.851, false)

Expected Results:	Actual Results:
false true true false false true true false false true true false	

Post-Conditions: None.

Table 60: Check for Walls

Test ID: 3.1.27	Module: SimulationInstance	Status: TBT

Get Angle from Coordinates

Description: Run the method which uses an x- and a y-coordinate to obtain the angle from that imaginary triangle.

Pass/Fail Conditions: This test is passed if the expected results are within 0.0001 of the actual results. Notably in the case where x = y = 0, the angle will be $\frac{3}{2}\pi$ which is not technically correct but that does not matter for this project.

Pre-Conditions: None

Inputs:
(1, 0)
(2, 1)
(0, 1)
(-1, 2)
(-1, 0)
(-1, -5)
(0, -1)
(2, -3)

Expected Results:	Actual Results:
0 0.463647609 $\frac{\pi}{2}$ 2.034443936 π 4.514993421 $\frac{3\pi}{2}$ 5.300391584	

Post-Conditions: None.

Table 61: Get Angle from Coordinates

Test ID: 3.1.28	Module: SimulationInstance	Status: TBT
Ball-Wall Collision		
Description: Runs the method which evaluates the resulting velocities from ball-wall collisions.		
Pass/Fail Conditions: This test is passed if the expected results are within 0.0001 of the actual results.		
Pre-Conditions: None Inputs: (5, true) (-1.2, false)		
Expected Results: -4.33 -1.2	Actual Results:	
Post-Conditions: None.		

Table 62: Ball-Wall Collision

Test ID: 3.1.29	Module: SimulationInstance	Status: TBT

Check if in Pocket

Description: Runs the method which checks whether the given coordinate would result in a ball being sunk into a pocket.

Pass/Fail Conditions: This test is passed if the expected results are equal to the actual results.

Pre-Conditions: None

Inputs:
(1, 0.5)
(0,0)
(0.06, 0.02)
(0, 0.921)
(0.03, 0.92)
(0.924,0)
(0.92, 0.02)
(0.924, 0.921)
(0.95, 0.921)
(1.848,0)
(1.84, 0.04)
(1.848, 0.921)
(1.84, 0.915)

Expected Results:	Actual Results:
false true false true false true false true false true false true false	

Post-Conditions: None.

Table 63: Check if in Pocket

TableState Tests

Test ID: 3.1.30	Module: TableState	Status: TBT
TableState Constructor Good Inputs		
Description: Builds a new TableState.		
Pass/Fail Conditions: This test is passed if the TableState is successfully created and stores the correct information.		
Pre-Conditions: None Input: A 16-by-2 array of doubles that hold the position of the balls		
Expected Results: A new TableState with 16 balls in positions corresponding to those passed in.	Actual Results:	
Post-Conditions: TableState has been created.		

Table 64: TableState Constructor Good Inputs

Test ID: 3.1.31	Module: TableState	Status: TBT
TableState Constructor Too Many Elements		
Description: Builds a new TableState with too many elements in the outer array.		
Pass/Fail Conditions: This test is passed if the TableState is not created.		
Pre-Conditions: None Input: A 17-by-2 array of doubles		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: TableState has not been created.		

Table 65: TableState Constructor Too Many Elements

Test ID: 3.1.32	Module: TableState	Status: TBT
TableState Constructor Not Enough Elements		
Description: Builds a new TableState with not enough elements in the outer array.		
Pass/Fail Conditions: This test is passed if the TableState is not created.		
Pre-Conditions: None Input: A 15-by-2 array of doubles		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: TableState has not been created.		

Table 66: TableState Constructor Not Enough Elements

Test ID: 3.1.33	Module: TableState	Status: TBT
TableState Constructor Elements Too Small		
Description: Builds a new TableState with not enough elements in one of the inner arrays.		
Pass/Fail Conditions: This test is passed if the TableState is not created.		
Pre-Conditions: None Input: A 16-by-1 array of doubles		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: TableState has not been created.		

Table 67: TableState Constructor Elements Too Small

Test ID: 3.1.34	Module: TableState	Status: TBT
TableState Constructor Elements Too Large		
Description: Description: Builds a new TableState with too many elements in one of the inner arrays.		
Pass/Fail Conditions: This test is passed if the TableState is not created.		
Pre-Conditions: None Input: A 16-by-3 array of doubles		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: TableState has not been created.		

Table 68: TableState Constructor Elements Too Large

Test ID: 3.1.35	Module: TableState	Status: TBT
TableState Deep Copy		
Description: Runs the method which returns a deep copy of the TableState passed in.		
Pass/Fail Conditions: This test is passed if the array of Balls returned have the same values but are not the same Objects.		
Pre-Conditions: A TableState exists in memory. Input: None.		
Expected Results: An array of Balls that have the same positions as those in the TableState.	Actual Results:	
Post-Conditions: None.		

Table 69: TableState Deep Copy

5.1.2 PC VR Program

Test ID: 3.2.1	Module: PC VR test 1	Status: PASS
Ball Recognition and colour		
Description: An image of the table is provided and the results of the VR		
Pass/Fail Conditions: The measured positions are within 5 millimetres of the actual positions.		
Pre-Conditions: None. Input: Image of table		
Expected Results: (1350, 510) (390, 450) (1350, 460) (1300, 490) (1350, 410) (1400, 540) (1460, 510) (1400, 430) (1400, 480) (1300, 430) (1450, 350) (1250, 460) (1800, 60) (1450, 460) (1450, 400) (1450, 560)	Actual Results:	
Post-Conditions: Results are written to TableState.csv		

Table 70: Test Title

Test ID: 3.2.2	Module: PC VR test 2	Status: PASS

Ball Recognition and colour

Description: An image of the table is provided and the results of the VR

Pass/Fail Conditions: The measured positions are within 5 millimetres of the actual positions.

Pre-Conditions: None.

Input: Image of table

Expected Results:	Actual Results:
(690, 410) (1150, 290) (1060, 540) (970, 440) (1140, 440) (1140, 430) (470, 570) (310, 350) (-1, -1) (-1, -1) (-1, -1) (-1, -1) (-1, -1) (-1, -1) (-1, -1) (-1, -1)	

Post-Conditions: Results are written to TableState.csv

Table 71: Test Title

5.1.3 μC Program

Certain functions of this program (such as functionality of sensors) are tested in the electrical section and so will NOT be tested again here.

Test ID: 3.3.1	**Signal steps for X Motion**	Status: TBT
Description: Generates the signals to move the machine to the destination.		
Pass/Fail Conditions: System is capable is tracking an arbitrary number of steps upon request of motion (concurrently with all axes and rotations).		
Pre-Conditions: None. Input: Motion request in X axis (system repeats arbitrary motion 10 times).		
Expected Results: After testing cycle the system step count should match theoretical count.	Actual Results:	
Post-Conditions: None.		

Table 72: Signal steps for X Motion

Test ID: 3.3.2	**Signal steps for Y Motion**	Status: TBT
Description: Generates the signals to move the machine to the destination.		
Pass/Fail Conditions: System is capable is tracking an arbitrary number of steps upon request of motion (concurrently with all axes and rotations).		
Pre-Conditions: None. Input: Motion request in X axis (system repeats arbitrary motion 10 times).		
Expected Results: After testing cycle the system step count should match theoretical count.	Actual Results:	
Post-Conditions: None.		

Table 73: Signal steps for Y Motion

Test ID: 3.3.3	**Signal steps for Rotational Motion**	Status: TBT
Description: Generates the signals to rotate the machine to the destination angle.		
Pass/Fail Conditions: System is capable is tracking an arbitrary number of steps upon request of motion (concurrently with all axes and rotations).		
Pre-Conditions: None. Input: Motion request in X axis (system repeats arbitrary motion 10 times).		
Expected Results: After testing cycle the system step count should match theoretical count.	Actual Results:	
Post-Conditions: None.		

Table 74: Signal steps for Rotational Motion

Test ID: 3.3.4	**Calculation of Steps Required**	Status: TBT
Description: A target location will be used to compute the required signals to move the machine to that location.		
Pass/Fail Conditions: Is capable of converting between linear or rotational displacement and number of steps		
Pre-Conditions: None. Input: Linear or rotational distance (repeat this test with a vareity of values (both positive and negative)).		
Expected Results: Ouput to console actual number of steps corresponding to the theoretical values.	Actual Results:	
Post-Conditions: The machine should not have moved or be moving.		

Table 75: Calculation of Steps Required

Test ID: 3.3.5	**Signal for Pneumatic Extension**	Status: TBT
Description: Generates the signals to fire the piston as appropriate.		
Pass/Fail Conditions: System powers on 12VDC signal to power pneumatic valve necessary for piston extension.		
Pre-Conditions: None. Input: System request signal for pneumatic piston extension.		
Expected Results: 12VDC detected and at output (use multimeter or oscilloscope) and output to console.	Actual Results:	
Post-Conditions: None.		

Table 76: Signal for Pneumatic Extension

Test ID: 3.3.6	**Signal for Pneumatic retraction**	Status: TBT
Description: Generates the signals to retract the piston to its default position.		
Pass/Fail Conditions: System powers on 12VDC signal to power pneumatic valve necessary for piston retraction.		
Pre-Conditions: None. Input: System request signal for pneumatic piston retraction.		
Expected Results: 12VDC detected and at output (use multimeter or oscilloscope) and output to console.	Actual Results:	
Post-Conditions: None.		

Table 77: Signal for Pneumatic Extension

Test ID: 3.3.7	**Signal Steps for Rotational Motion of Air Flow Controller**	Status: TBT
Description: Generates the signals to rotate the air flow valve		
Pass/Fail Conditions: System is capable is tracking an arbitrary number of steps upon request of motion (concurrently with all axes and rotations).		
Pre-Conditions: None. Input:rotational distance (repeat this test with a variety of values (both positive and negative)).		
Expected Results: After testing cycle the system step count should match theoretical count.	Actual Results:	
Post-Conditions: None.		

Table 78: Signal Steps for Rotational Motion of Air Flow Controller

5.2 System Tests

Test ID: 4.1	Module: System	Status: TBT
Aligned Shot		
Description: The user will press the "Take Shot" button, the system will go through its whole process and then shoot the cue ball to sink the target ball.		
Pass/Fail Conditions: This test is passed if the target ball is sunk by the machine 50% of the time and the shot should be made within 90 seconds (as per the *Summary and Goals* document).		
Pre-Conditions: Machine must not be currently moving or taking a shot. There are three balls on the table, the cue ball, the target ball, and the eight ball. The cue ball, target ball, and one of the pockets are aligned near perfectly along an imaginary line. The eight ball is not in a position to interfere with motion of the balls along that line. Input: *Take Shot* button pressed.		
Expected Results: Only the target ball should be sunk.	Actual Results:	
Post-Conditions: The eight ball should remain on the table. The target ball should be sunk. There are no requirements for the cue ball, but bonus points if it remains on the table.		

Table 79: Aligned Shot

Test ID: 4.2	Module: System	Status: TBT
Angled Shot		
Description: The user will press the "Take Shot" button, the system will go through its whole process and then shoot the cue ball to sink the target ball.		
Pass/Fail Conditions: This test is passed if the target ball is sunk by the machine 50% of the time and the shot should be made within 90 seconds (as per the *Summary and Goals* document).		
Pre-Conditions: Machine must not be currently moving or taking a shot. There are three balls on the table, the cue ball, the target ball, and the eight ball. There should be a shot that can be made with a modest angle that will sink the target ball. The eight ball is not in a position to interfere with expected motion of the balls. Input: *Take Shot* button pressed.		
Expected Results: Only the target ball should be sunk.	Actual Results:	
Post-Conditions: The eight ball should remain on the table. The target ball should be sunk. There are no requirements for the cue ball, but bonus points if it remains on the table.		

Table 80: Angled Shot

Test ID: 4.3	Module: System	Status: TBT
Shot Cancelled Before Motion		
Description: The user will press the "Take Shot" button, the system will begin going through its process. Before motion begins, the "Cancel" button will be pressed. The system will then cease its prior execution.		
Pass/Fail Conditions: This test is passed if the machine does not move.		
Pre-Conditions: None. Input: *Take Shot* button pressed, Then *Cancel* button pressed before machine moves.		
Expected Results: The machine should cancel the instruction and not move.	Actual Results:	
Post-Conditions: The machine should not have moved or be moving.		

Table 81: Shot Cancelled Before Motion

Test ID: 4.4	Module: System	Status: TBT
Shot Cancelled During Motion		
Description: The user will press the "Take Shot" button, the system will begin going through its process. After motion begins, the "Cancel" button will be pressed. The system will then cease motion.		
Pass/Fail Conditions: This test is passed if the machine ceases movement within 2 seconds.		
Pre-Conditions: None. Input: *Take Shot* button pressed, Then *Cancel* button pressed while machine is moving.		
Expected Results: The machine should cease movement.	Actual Results:	
Post-Conditions: The machine should not be moving.		

Table 82: Shot Cancelled During Motion

Test ID: 4.5	Module: System	Status: TBT
Move Request (To Zero X-Coordinate)		
Description: The user will press the “Move” button. The machine will then move to the zero x-coordinate.		
Pass/Fail Conditions: The machine moves to the zero x-coordinate within 20 seconds.		
Pre-Conditions: Machine’s y-rail is located closer to the large x-coordinate. Input: *Move* button pressed		
Expected Results: The machine should move to the zero x-coordinate of the table.	Actual Results:	
Post-Conditions: The machine should be located at the zero x-coordinate.		

Table 83: Move Request (To Zero X-Coordinate)

Test ID: 4.6	Module: System	Status: TBT
Move Request (To Largest X-Coordinate)		
Description: The user will press the “Move” button. The machine will then move to the largest x-coordinate.		
Pass/Fail Conditions: The machine moves to the largest x-coordinate within 20 seconds.		
Pre-Conditions: Machine’s y-rail is located closer to the zero x-coordinate. Input: *Move* button pressed		
Expected Results: The machine should move to the largest x-coordinate of the table.	Actual Results:	
Post-Conditions: The machine should be located at the largest x-coordinate.		

Table 84: Move Request (To Largest X-Coordinate)

Test ID: 4.7	**Shot Power Modification**	Status: TBT
Description: Users should not be able to modify system to perform unsafe actions such as setting the power of a shot beyond a certain safe value. The test will attempt to make the system do just that.		
Pass/Fail Condition: This test is passed if the user cannot modify the power the shot beyond system parameters.		
Pre-Conditions: None. Input: User attemtpts to take a shot with power outside of system parameters.		
Expected Results: System does not take a shot at that level of force.	Actual Results:	
Post-Conditions: None.		

Table 85: Shot Power Modification

Test ID: 4.8	Module: System	Status: TBT
Check For Political Correctness		
Description: Colleagues will be asked whether the machine has any direct references to any religious or political groups.		
Pass/Fail Conditions: All interviewees agree that there are no direct references to any religious or political groups.		
Pre-Conditions: None. Input: 20 colleagues will be asked to give their opinion on whether the system created has no direct reference to any religious or political groups.		
Expected Results: Colleagues decide that there are no direct references to any religious or political groups.	Actual Results:	
Post-Conditions: None.		

Table 86: Check For Legality and Political Correctness

Test ID: 4.9	Module: System	Status: TBT
Assessment of Durability		
Description: The machine will play through 3 games.		
Pass/Fail Conditions: The machine is still in full functional order.		
Pre-Conditions: None. Input: The machine will be used to play 3 full games.		
Expected Results: The machine is still fully functional.	Actual Results:	
Post-Conditions: None.		

Table 87: Assessment of Durability

6 Summary of Results

This section will be completed once the first version of the system is completed and all tests can be run.