Lab III - PolyMorpher Prototype Product Test Plan

Team Silver

Old Dominion University

CS411W

Professor Thomas Kennedy

2 April 2018

Version 1

Author Note

Team Silver, Department of Computer Science, Old Dominion University.

This research was done under the supervision and guidance of Thomas Kennedy.

Correspondence concerning this article should be addressed to Team Silver, Department of Computer Science, Old Dominion University, Norfolk, VA 23529.

Contact: http://www.cs.odu.edu/~411silver/

Table of Contents

Lab 3 - PolyMorpher Product Test Plan

1 Objectives

The purpose of this Test Plan is to test the operation and performance of the PolyMorpher Prototype. It will be testing the six major test categories of PolyMorpher - General Level, UI, Sound & Animation, OOP Puzzle, Script, and Performance. Each category covers related aspects of the prototype. (Kevin Santos)

2 References

Team Silver. (2018, February 26). Lab 1. In *docs.google.com*.

Team Silver. (2018, February 28). Lab 2. In *docs.google.com*. (Kevin Santos)

3 Test Plan

The test plan of the prototype will cover the tests for the robustness of the game, testing that the player can play the game without interruptions, and testing that all the functionalities of the UI/UX design work appropriately. In addition, the testing approach, the identification of tests, the test schedule, fault reporting and data recording, resource requirements, test environment, and test responsibilities will be covered in the test plan of the prototype. (Kevin Santos)

3.1 Testing Approach

Performance of the prototype will be verified through the following categories of system and gameplay testing:

1. **General Level** tests verify that each level for each scene contains proper functionalities including the in-game menu.

2. **UI** tests verify that each game level displays the proper UI. The UI for every level has to provide the necessary tools and provide user friendliness to the player.

3. **Sound & Animation** tests will verify that the sounds and animations properly work when the game is running.

4. **OOP Puzzle** tests verify that the each OOP puzzle is teaching the appropriate lesson for each level.

5. **Script** tests verify that the scripts given to the player to complete actually follow the C# convention.

6. **Performance** tests verify that the game performs at optimal level on the player's machine.

(Kevin Santos)

3.2 Identification of Tests

Each test will be identified by which of the six categories it corresponds to and by the test case number that it has been assigned. (Peter Riley)

Procedure ID	**Category**	**Test Case**	**Description**	**Objective**
1	General Level	1.1	Player Input Test	To verify that the player keyboard input is working as intended.
		1.2	Character Boundary Test	To verify the sprite tiles of the UI displays as intended.
2	UI	2.1	Button UI Test	To verify that the API Book, Settings, and Reset buttons are working an intended
		2.2	GameObject Highlights Test	To verify that every GameObject in the game is highlighted as intended

		2.3	TextBook UI Test	To display the "Textbook" learning curriculum before and after each level.
3	Sound & Animation	3.1	Player Sounds and Animation	Test to see if the player's animations act as intended.
		3.2	NPC Sounds and Animations	Test to see if the NPC's animations act as intended
4	OOP Puzzle	4.1	Inheritance level test	Test to ensure that the inheritance level's obstacle course and superclasses function as intended
		4.2	Abstraction level test	Test to make sure that abstraction puzzle works as intended without any major bugs
		4.3	Encapsulation level test	To verify that gameplay within the Encapsulation level is as intended
		4.4	Polymorphism level test	To verify that gameplay within the Polymorphism level is as intended
		4.5	Tutorial Mechanics level tests	Test to make sure that tutorial level for the mechanics game works as intended without any major bugs
		4.6	Coding Tutorial level test	Test to ensure that the coding tutorial level and related API instructions function as intended

5	Script	5.1	Script Execution	Verify that the script execution sequence executes through all steps without failure, while checking compilation success of custom player scripts.
6	Performance	6.1	Performance Test	To verify the the game works on the chosen operating systems and minimum system requirements

3.3 Test Schedule

A full test will take an approximate time of fifty minutes to complete. (Colten Everitt)

Start Time (hour:min)	**Duration (mins)**	**Test Case**	**Description**	**Comments**
0:00	3	1.1	Player Input Test	General Level Testing Part 1 & Sounds Part 1
0:03	3	1.2	Character Boundary Test	General Level Testing Part 2
0:06	3	2.1	Button UI Test	UI Tests Part 1
0:09	3	2.2	GameObject Highlights Test	UI Tests Part 2
0:12	3	2.3	TextBook UI Test	UI Tests Part 3
0:15	3	3.1	Player Sounds & Animation	Sounds Part 2 & Animations Part 1
0:18	3	3.2	NPC Sounds & Animation	Sounds Part 3 & Animations Part

				2
0:21	4	4.1	Inheritance Level Test	Inheritance Puzzle
0:25	4	4.2	Abstraction Level Test	Abstraction Puzzle
0:29	4	4.3	Encapsulation Level Test	Encapsulation Puzzle
0:33	4	4.4	Polymorphism Level Test	Polymorphism Puzzle
0:37	4	4.5	Tutorial Mechanics Level Test	Tutorial Mechanics Puzzle
0:41	4	4.6	Coding Tutorial Level Test	Tutorial Code Puzzle
0:45	3	5.1	Script Execution Test	Script Execution
0:48	2	6.1	Performance Test	Performance Requirements

3.4 Fault Reporting and Data Recording

The results of system and gameplay tests should be recorded or displayed in an Excel Spreadsheet or Google Document after the test is run; the results of the test will be recorded. If what occurs during the test is what is expected from the given test case, then the test will be reported as a success. Otherwise, the test will be reported as a failure. Tests requiring user feedback responses will be recorded by hand by a member of Team Silver during the testing demonstration. (Tyler Johnson)

3.5 Resource Requirements

For the testing procedure, certain requirements must be made available. A computer running Mac OSX or Windows and the latest version of Unity will be required to run the test. The computer must also meet the system requirements for the game, such as: an Intel i3 processor, 4GB of RAM, and at least 5GB of hard drive space. Microsoft Excel or Google Documents will also be needed to record the results of the test. An active connection to the Internet will only be needed to download the game from the PolyMorpher website. No Internet connection is required to play the game. (Tyler Johnson)

3.6 Test Environment

Software tests will be run on a computer that is running the latest version of Unity. For example, they could be conducted remotely on the Team Silver virtual Windows 7 machine. Tests requiring user feedback will be conducted in a recitation classroom in the Gornto Telecommunications Building at Old Dominion University and will be recorded live using WebEx. (Tyler Johnson)

3.7 Test Responsibilities

Each member of Team Silver will perform distinct roles necessary for the completion of the tests. (Cole Everitt)

Team Member	Role	Responsibilities
Batten, Casey	Tester	Performs the tests
Dang, Daniel		
DeArce, Nathaniel		
Tuckson, Matthew		

Everitt, Colten	Documenter	Documents changes and comments about PolyMorpher
Santos, Kevin	Speaker	Explains the results
Riley, Peter		
Johnson, Tyler	Technical Advisor	Answers questions about PolyMorpher software functionality
Stokes, Joel		
Tuckson, Matthew	Test Manager	Lead test demonstration team

4 Test Procedures

The procedures for each of the test cases are included in this section. The test cases have been made so that they cover each of the requirements that were covered in the prototype. These details include: a list of the requirements that the test fulfills, a description of the test and its purpose, the author of the test, and its individual test steps and expected outputs. (Tyler Johnson)

4.1 General Level Tests

This suite of tests will cover varying aspects of gameplay. These tests will cover test case topics that should be tested in every level such as: player movement, player boundaries, and level boundaries. These test cases are meant to ensure gameplay functionality and ensure that the levels are working as expected.

Test Category: Systems / Play		**Description:** Test the player movement	
Test Case: 1.1	**Case Name:** Player Input Test	**Version:** 1.0	**Written By:** Daniel Dang
Requirements Fulfilled: 3.2.1.1, 3.2.2.1		**Purpose:** To verify that the character input is working as intended.	

Setup Conditions: ● Load scene ● Have keyboard connected				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	The player presses the left arrow key			The character shall move to the left
2	The player presses the right arrow key			The character shall move to the right
3	The player presses the up arrow key			The character shall do a single jump animation
4	The player releases the left arrow key			The character stops moving left
5	The player releases the right arrow			The character stops moving right
6	The character traverses the map			The camera shall follow the character with a slight delay

Test Category: Systems / Play		**Description:** Test the boundaries of each level		
Test Case: 1.2	**Case Name:** Character Boundary Test	**Version:** 1.0	**Written By:** Daniel Dang	
Requirements Fulfilled: 3.2.5.1-3.2.5.2		**Purpose:** To verify the sprite tiles of the UI displays as intended.		
Setup Conditions: ● Run the executable ● Load the first scene of the game				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Traverse the character to edge of the map			The character shall stop moving at the sprite tile that represents the boundary.

2	Move the character to the left			The character should be able to move freely within the background tiles until it reaches the edge.
3	Move the character to the right			The character should be able to move freely within the background tiles until it reaches the edge.
4	Repeat steps 1-3 for each level			The same results are expected for each level

4.2 User Interface (UI) Tests

This suite of tests will cover the user interface portions of the game. These tests will specifically cover the Morphing User Interface and the API Book Interface. These test cases are meant to ensure that these user interfaces are functioning as expected.

Test Category: Systems		**Description:** Verify that the API Book, Settings, Reset buttons, and Signs are working as intended		
Test Case: 2.1	**Case Name:** Button UI Test	**Version:** 1.0	**Written By:** Nathaniel DeArce	
Requirements Fulfilled: 3.1.1.1 - 3.1.1.2 , 3.1.1.5, 3.1.2.1-3.1.2.2, 3.1.2.6		**Purpose:** To verify that the API Book, Settings, Reset buttons, and Signs are working an intended		
Setup Conditions: ● Run the executable				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Click on the executable			The game should load and begin running without issue

2	Click on the API Book button		This will be found at the top-right corner of the screen	The custom API should display with recommended code snippets for the player
3	Close the custom API display			The custom API should no longer be visible
4	Click on the Settings button		This will be found in some region of the screen	The Settings menu should display
5	Close the Settings menu			The Settings menu should no longer be visible
6	Make any change to the current scene		Changes should include moving the PC, moving any object within the scene, and editing any script within the scene	The current scene should be noticeably different from its original state
7	Click on the Reset button		This will be found in some region of the screen	The current scene should return to its original state
8	Click on a Sign GameObject		Level traversal may be necessary to reach a sign	A text box displaying the contents of the sign should be displayed
9	Inspect the text box contents		A reference will be available that explains what each sign should display	The contents should be correct according to the reference
10	Repeat steps 8-9 for each Sign in each scene			All signs should display what is intended

Test Category: Systems		**Description:** Verify that every GameObject is highlighted as intended		
Test Case: 2.2	**Case Name:** GameObject Highlights Test	**Version:** 1.0	**Written By:** Nathaniel DeArce	
Requirements Fulfilled: 3.1.1.3 - 3.1.1.4		**Purpose:** To verify that every GameObject in the game is highlighted as intended		
Setup Conditions: ● Run the executable ● Load the first scene of the game				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Mouse over a GameObject		GameObjects are any objects within the scene	The GameObject should be highlighted or remain the same
2	Click on the GameObject		Repeat steps 1 - 2 for another GameObject if the GameObject was not highlighted. Proceed to Step 3 if the GameObject was highlighted.	Results should be as follows: Highlighted: A text editor should open displaying the contents of the associated script Not highlighted: Nothing should happen
3	Inspect the displayed UI			Green highlight: The Compile and Reset button should be visible Red highlight: Only the text editor should be visible
4	Click on the text editor			Green: It should be possible to edit the script Red: It should not be possible to edit the script

5	Inspect the contents of the text box		A reference will be provided that details the content of each script	The script should match the provided reference
6	Return to step 1		Proceed to step 7 once all GameObjects have been tested	All GameObjects within the scene should have been tested for the behavior denoted in steps 2 - 4.
7	Load the next scene and repeat steps 1-6			All scenes should be tested

Test Category: Systems / Play		**Description:** Test the prelevel and postlevel textbook functionality		
Test Case: 2.3	**Case Name:** TextBook UI Test	**Version:** 1.0	**Written By:** Daniel Dang	
Requirements Fulfilled: 3.1.3.3		**Purpose:** To display the “Textbook” learning curriculum before and after each level.		
Setup Conditions: ● Run the executable ● Load the scene of every level				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Load the scene			The game shall load the Textbook.
2	Click on the X button to exit the text book and start the game			The textbook shall disappear and the player can begin the game.
3	Click on the textbook icon to reactivate the textbook			The textbook should appear again for the player to review.

4	The player finishes the game level			The textbook appears again with new text to advance the players progress.
5	Repeat steps 1 - 4 for each level in the game.			All textbook popups should work as intended.

4.3 Sound & Animation Tests

This suite of tests will cover the sounds and animations of the game. These tests will check if the animations of the player and other non-player characters are running and functioning correct. The sound effects and music of the game will also be tested to ensure that the sounds are being played and at an appropriate volume.

Test Category: Systems / Play		**Description:** Test player animations and sounds		
Test Case: 3.1	**Case Name:** Player Animation	**Version:** 1.0	**Written By:** Matthew Tuckson	
Requirements Fulfilled: 3.2.1.2 - 3.2.1.4		**Purpose:** Test to see if the player's animations act as intended.		
Setup Conditions: ● Load the Abstraction Scene				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Hold the left arrow key to move the player to the left.			A walking animation should begin playing as well as walking sounds.
2	Release the left arrow key.			Walking animation should stop playing as well as walking sounds.

3	Move to the right side of the scene.. Click on the placeholder object.			An attacking animation should occur and stop while you enter the ingame IDE. During the animation a corresponding attack sound should occur.
4	Fall through the hole in the ground.		This should happen every time the player dies. If it works in one case it will work in every case.	A death animation should play on the player and the scene should then be reloaded. A death sounds should occur during this time as well.

Test Category: Systems / Play		**Description:** NPC animations and sounds tests		
Test Case: 3.2	**Case Name:** NPC Animation	**Version:** 1.0	**Written By: Matthew Tuckson**	
Requirements Fulfilled: 3.2.3.1-3.2.3.3		**Purpose:** Test to see if the NPCs' animations act as intended.		
Setup Conditions: ● The following steps will need to be completed for each NPC in the game ● Progress to the NPC being tested				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Move to where you can see the NPC, but far enough so that it will not interact with you.			If the NPC moves, there should be a movement animation and sounds for the NPC. If it does not move, there will not be an animation or sound.

2	Move to where you touch the NPC.			If the NPC is an enemy, an attacking animation should play and a corresponding sound should occur.
3.	If the NPC is killable, fulfill the conditions to kill the NPC.		An NPC is killable if in tests 8-13 it is said to die. The conditions to kill the NPC will also be defined in those specific scripts.	The NPC's death animation will appear if it is killable. The NPC's death sound will occur at the same time.

4.4 OOP Puzzle Tests

This suite of tests will cover the various puzzles for each level of the game. These tests will ensure that each puzzle for each level is clearable and functions as expected. These test will cover each levels' unique puzzle and ensure that the player can solve with information that is given.

Test Category: Systems/Play		**Description:** Inheritance level test	
Test Case: 4.1	**Case Name:** Inheritance Test	**Version:** 1.0	**Written By: Casey Batten**
Requirements Fulfilled: 3.3.1.1-3.3.1.2		**Purpose:** Test to ensure the Inheritance level and all related level assets work as intended.	
Setup Conditions: ● **Open a web browser** ● **Go to the PolyMorpher website under the 'Download' tab** ● **Click on the 'Download' button to download the executable file of the game** ● **Run the executable file of the game to start playing the game**			

Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Load the Inheritance level			The level should load with all starting assets in their proper/expected locations
2	Signs with information should be selectable			Signposts containing level information and instructions should all be interactable and provide the relevant information
3.	Vehicle Class and related functions should be visible under the Suggested API			Examples from the Vehicle Class and the related functionality for the available components of the vehicle should be in under the Suggested API and executable within player code
4.	Vehicle should be present at start of level			The base vehicle body should be at the start of the level, with the ability to select it for code addition and mount it for riding
5.	Vehicle should adhere to expected operation requirements			The vehicle should only be able to move if it has an engine, fuel, and some method of traversal via an attached vehicle part (ex: wheels)
6.	Obstacles should behave as expected			The various obstacles should prevent the vehicle/players progress unless the player builds a vehicle capable of overcoming them (ex: wheels to drive over a ramp to jump a spike pit)

Test Category: Systems / Play		**Description:** Abstraction level tests		
Test Case: 4.2	**Case Name:** Abstraction	**Version:** 1.0	**Written By:** Matthew Tuckson	
Requirements Fulfilled: 3.3.2.1 - 3.3.2.3		**Purpose:** Test to make sure that abstraction puzzle works as intended without any major bugs		
Setup Conditions: ● Load the Abstraction Scene				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Load or Progress the Abstraction Scene			The scene should load without issue
2	Click on each sign with a '?'.		The # of signs is currently undecided, however each should act the same.	A message should appear describing what you need to do.
3	Traverse to the right side of the room and click on each plant.		There are a total of three plants, so 3 clicks total in this step.	For each plant, an icon should appear indicating you have received that plant
4	Click on the door to the far left in the scene			The player should be transported to a room with a campfire and a plant should leave the player.
5	Click on the plant.			The ingame IDE should open.
6	Click on the API button (purple book) on the top left of the screen.			The ingame API should open.

7	Move the plant to the identified location via scripting (the exact location is not defined yet).			An icon An icon should appear indicating you have received the altered plant
8	When in the room with a campfire, click on the only door.			The player should be transported back to where they were prior to step 4 in the test.
9	Repeat steps 4 through 8 for the middle door in the scene.		.	The player will be transported to a room with ice and a similar interaction will occur as steps 4-8.
10	Repeat steps 4 through 8 for the right door in the scene.			The player will be transported to a room with water and a similar interaction will occur as steps 4-8.
11	Go to the far right side of the scene. Click the placeholder object and attach a script to it that freezes the arrow. (Exact script is not defined yet).			Ingame IDE opens allowing you to create the script.
12	Click on each altered plant and reference the script you just made in each of their scripts. (Exact implementation not defined yet).			No errors should occur after compiling the scripts.

13	Move each plant into the hole so that they fall into the cauldron.			The arrow should freeze when the plant falls. It starts moving after the plant hits the cauldron. After the third plant hits the cauldron, then Orc should die.
14	Fall through the hole and click the door.			The next scene should open up. (Order of scenes not decided yet).

Test Category: Systems / Play		**Description:** Verify that gameplay within the Encapsulation level is as intended		
Test Case: 4.3	**Case Name:** Encapsulation	**Version:** 1.0	**Written By:** Nathaniel DeArce	
Requirements Fulfilled: 3.3.3.1 - 3.3.3.2		**Purpose:** To verify that gameplay within the Encapsulation level is as intended		
Setup Conditions: ● Load the Encapsulation scene				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Load or progress to the Encapsulation scene			The scene should load without issue
2	Reach a “flying eyeball creature” (FEC)		Level traversal using a predetermined solution may be necessary to complete this step	The player should now be positioned just before a FEC
3	Attempt to progress past the FEC normally		This should be done without solving the puzzle	Attempt should fail and the required object to continue is returned to its original position

4	Attempt to progress past the FEC after attempting a wrong solution		Predetermined failed solutions will be used	Attempt should fail and the required object to continue is returned to its original position
5	Attempt to progress past the FEC after attempting a correct solution		Predetermined correct solutions will be used	Attempt should succeed and allow the player to continue

Test Category: Systems / Play		**Description:** Verify that gameplay within the Polymorphism level is as intended		
Test Case: 4.4	**Case Name:** Polymorphism	**Version:** 1.0	**Written By:** Nathaniel DeArce	
Requirements Fulfilled: 3.3.4.1 - 3.3.4.2		**Purpose:** To verify that gameplay within the Polymorphism level is as intended		
Setup Conditions: ● Load the Polymorphism Scene				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Load or progress to the Polymorphism scene			The scene should load without issue
2	Move around and inspect the scene		Avoid contact with the FloorButton	A Read-only wooden box should be present A tower of 3 wooden boxes and a moveable block should be present. The tower of wooden boxes should be immovable.

3	Move over the floor button			The FloorButton should depress and activate a fire animation
4	Mouse over the fire animation			This animation should be a read-only object and display the red highlight
5	Move off of the floor button			The fire animation should stop
6	Continue past the corridor and inspect the scene			A tower of immovable wooden boxes and metal boxes should be present. A small fire animation should be playing below them. A button should be embedded in the ceiling.
7	Attempt to progress past the tower without modifying anything			The player should not be able to progress past the tower
8	Modify the fire animation to burn the wooden boxes		This should be done without activating the button in the ceiling. Code to do this will be provided.	The wooden boxes should be destroyed and the metal boxes should fall to the ground
9	Press the F1 key			The scene should reset to its initial state

10	Return to the area in step 6			Refer to step 6
11	Using the moveable block from the first area, press the ceiling button		This should be done without modifying the fire. Code to do this will be provided	A read-only ice animation should play and "freeze" the metal boxes. The metal boxes should turn blue.
12	Repeat step 8			The fire animation should destroy the wooden boxes, and the metal boxes should remain in place
13	Progress forward and inspect the scene			There should be a stack of wooden boxes and metal boxes
14	Attempt a failing solution		Failing solutions will be provided. Use the Reset key as necessary	The player should not be able to climb the wall and progress.
15	Attempt a successful solution		Successful solutions will be provided	The metal boxes should be frozen in place to make staircase, the wooden boxes should be destroyed by the fire

Test Category: Systems / Play		**Description:** Tutorial Mechanics level tests	
Test Case: 6.5	**Case Name:** Tutorial Mechanics	**Version:** 1.0	**Written By:** Matthew Tuckson
Requirements Fulfilled: 3.3.45.1 - 3.3.5.10		**Purpose:** Test to make sure that tutorial level for the mechanics of the game works as intended without any major bugs.	

Setup Conditions: ● Load the Tutorial Mechanics Scene				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Load or progress to the Tutorial Mechanics Scene			The scene should load without issue. A tutorial box should appear on the screen giving movement instructions.
2	Click the next button.		Instructions are numbered.	The next instruction should appear.
3	Click the back button.			The previous instruction should appear.
4.	Click on the green block.			The ingame IDE should appear.
5.	Click on the purple API button on the top left of the screen.			The ingame API should appear.
6.	Move the block so that it falls onto the raised platform via scripting.			The lock on the door should disappear.
7.	Click on the door.			The next scene should open up. (Order of scenes not decided yet).

Test Category: Systems/Play		**Description:** Coding Tutorial level test		
Test Case: 6.6	**Case Name:** Coding Tutorial Test	**Version:** 1.0	**Written By: Casey Batten**	
Requirements Fulfilled: 3.3.6.1-3.3.6.4		**Purpose:** Test to ensure the Coding Tutorial level and all related level assets work as intended.		
Setup Conditions: ● Open a web browser ● Go to the PolyMorpher website under the 'Download' tab ● Click on the 'Download' button to download the executable file of the game · Run the executable file of the game to start playing the game				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Load the Coding Tutorial level			The level should load with all starting assets in their proper/expected locations.
2	Signs with information should be selectable			Signposts containing level information and instructions should all be interactable and provide the relevant information.
3.	Level-specific API entries detailing definitions for Game Design terms and functions			The API Book should feature custom entries under the Suggested API option that includes specific entries that detail game design terminology and functionality such as GameObjects, transforms, hitboxes, etc.

4.	Box objects with various features attached and detached			Several box objects that the player must manipulate to exit the level, test that there is one without any components, one with a boxcollider component, and one with a rigidbody component.
5.	Locked door preventing player from exiting the level			The door at the end of the level should be locked, only unlocking when a series of switches have been activated allowing the player to leave.
6.	Switches for the boxes to be moved to			The series of switches will require the boxes to interact with them in certain capacities and must be able to check that the requirements for “passing” the player have been completed.

4.5 Script Tests

This suite of tests will cover the script execution algorithm of the game. These tests will specifically cover the script execution algorithm and as a byproduct will cover the Morphing User Interface. These test cases are meant to ensure that the player can execute scripts that are semantically and syntactically correct.

[Space left Intentionally]

Test Category: Systems		**Description:** Save, build and compile a script created by the player for the purpose of changing the behavior of objects in the scene.			
Test Case: 5.1	**Case Name:** Script Execution	**Version:** 1.0	**Written By: Casey Batten**		
Requirements Fulfilled: 3.1.2.4, 3.1.2.7, 3.4.1.1-3.4.2.1, 3.7.1.1		**Purpose:** Verify that the script execution sequence executes through all steps without failure, while checking compilation success of custom player scripts.			
Setup Conditions: ● Open a web browser ● Go to the PolyMorpher website under the 'Download' tab ● Click on the 'Download' button to download the executable file of the game ● Run the executable file of the game to start playing the game					
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**	
1	Mouse over selectable object			The object should highlight green if editable and red if read only.	
2	Click on selectable object			The coding interface or read-only interface should open.	
3.	Add code to move object			Code should be able to be entered to change the objects position by X amount in the given direction on a 3D vector line axis.	
4.	Compile code			For editable objects, if the "Compile" button is clicked the script should run and the changes should be applied.	
5.	Error return			If the code errors out during	

				compilation it should return a pop-up window with the error information and line number for the error.
6.	Check that script saved on successful compilation			When the code successfully compiles the script should be saved as an "ObjectName.cs" file in the Streaming Assets folder of the game's file directory.

4.6 Performance Tests

This suite of tests will cover operating system and hardware compatibility. These tests will specifically cover Linux, Windows and Mac OSX since these are the three major operating systems. These tests will also evaluate the game's performance while running with the minimum system requirements such as an Intel i3 processor and 4GB of memory.

Test Category: Systems / Play		**Description:**		
Test Case: 6.3	**Case Name:** Performance Testing	**Version:** 1.0	**Written By:** Daniel Dang	
Requirements Fulfilled: 3.5.1.1 - 3.5.1.4		**Purpose:** To verify the the game works on the chosen operating systems and minimum system requirements		
Setup Conditions: ● Have a machine with the minimum system requirements ● Load the specified operating system ● Load FPS ● Run the executable				
Test Case Activity		**Pass/Fail**	**Comments**	**Expected Result**
1	Run the game executable in the Windows operating system			The game shall execute without issues and begin to play

2	Run the game executable in the Mac OS operating system			The game shall execute without issues and begin to play
3	Run the game executable in the Mac OS operating system			The game shall execute without issues and begin to play
4	Using a machine with the minimum system requirements, execute the game in Windows			The gameplay shall be playable and operable.
5	Using a machine with the minimum system requirements, execute the game in Mac OS			The gameplay shall be playable and operable.
6	Using a machine with the minimum system requirements, execute the game in Linux			The gameplay shall be playable and operable.

5 Traceability to Requirements

The traceability matrix highlights the relationship between the prototype requirements and the corresponding test cases. (Cole Everitt and Joel Stokes)

		Test Case ID															**# of Test Cases**
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1
Requirem	3.1.1.1			X													1
	3.1.1.2			X													1
	3.1.1.3				X												1

en t ID	3.1.1.4				X												1
	3.1.1.5			X													1
	3.1.2.1			X													1
	3.1.2.2			X													1
	3.1.2.3			X													1
	3.1.2.4														X		1
	3.1.2.5			X													1
	3.1.2.6			X													1
	3.1.2.7														X		1
	3.1.3.1				X												1
	3.1.3.2			X													1
	3.1.3.3					X											1
	3.2.1.1	X															1
	3.2.1.2	X					X										2
	3.2.1.3	X					X										2
	3.2.1.4	X					X										2
	3.2.1.5	X					X										2
	3.2.1.6	X					X										2
	3.2.1.7	X					X										2
	3.2.2.1	X															1
	3.2.3.1							X									1
	3.2.3.2							X									1
	3.2.3.3							X									1
	3.2.3.4							X									1

	3.2.3.5							X										1
	3.2.5.1		X															1
	3.2.5.2		X															1
	3.2.6.1	X																1
	3.3.1.1							X										1
	3.3.1.2							X										1
	3.3.2.1									X								1
	3.3.2.2									X								1
	3.3.2.3									X								1
	3.3.3.1										X							1
	3.3.3.2										X							1
	3.3.4.1											X						1
	3.3.4.2											X						1
	3.3.5.1												X					1
	3.3.5.2												X					1
	3.3.5.3												X					1
	3.3.5.4												X					1
	3.3.5.5												X					1
	3.3.5.6												X					1
	3.3.5.7												X					1
	3.3.5.8												X					1
	3.3.5.9												X					1
	3.3.5.10												X					1
	3.3.6.1														X			1

	3.3.6.2														X		1
	3.6.3.3														X		1
	3.3.6.4														X		1
	3.4.1.1														X		1
	3.4.1.2														X		1
	3.4.1.3														X		1
	3.4.2.1														X		1
	3.5.1.1															X	1
	3.5.1.2															X	1
	3.5.1.3															X	1
	3.5.1.4															X	1
	3.7.1.1														X		1
	3.7.2.3.1								X	X	X	X	X	X			6
	3.7.2.3.2								X	X	X	X	X	X			6
	3.7.2.3.3								X	X	X	X	X	X			6