Verification & Validation

Trouble Seekers

Rev 1

Prepared by
Group Name: Buttered Waffle

Andrew Raudys	**0973268**
Nathan MainVille	**0960351**
Matt Leduc	**0963077**
Sean McLellan	**0960351**

Instructor:	Dr. Carette
Course:	SFWR ENG 4GP6
Date:	Apr. 8, 2013

1. Menu Requirements

Test cases related to functional requirements have their requirement number specified as per the 4GP6 Requirements Specification Document

1.1 Start New Game

Verification:
The start new game button is available on the Main Menu located on the first screen of the game. When pressed, the player loads the first level of the game, and the game starts.
Validation:
This design implementation is sufficient for the user needs, as it allows the player to start the game.
Testing:
Testing this functionality just requires the player to press the “new game” button on the main menu, and wait for the game to load the first level within a reasonable amount of time (~1 minute). (**Requirement #1**)

1.2 Save Game Menu

Verification:
The player is currently unable to access a menu to save the game, see *Requirement 1.3 Save Game* for more information.
Validation:
The player needs to be able to access a menu to save the game, see *Requirement 1.3 Save Game* for more information.
Testing:
The player needs to be able to access a menu to save the game, see *Requirement 1.3 Save Game* for more information.

1.3 Save Game

Verification:
The player is currently unable to save the game, it is currently one of our highest priorities post rev 1 since the ability to save the game is crucial to any game in this generation. Currently this requirement is not met as the player cannot select the save option from the main menu of the game.
Validation:
The player needs to be able to save their progress at any point in time in order to pick up where they left off at any time, our game is currently lacking this feature and therefore fails this user requirement, we are unsure if it will be implemented.
Testing:
No testing could be done since saving the game is not yet implemented. (**Requirement #3**)

1.4 Load Game Menu

Verification:

The load game menu is not currently working in game, we have completed the Flash file which will be used to provide functionality to the menu but the buttons are not yet implemented in game. The requirement is therefore unfinished as we need to make sure the player can select the available load game options.

Validation:

The user needs to be able to select their load files from the menu in order to resume the game from that point, the menu needs to have buttons available to click for functionality. The user requirements is unfinished since the menu is created simply not functional yet.

Testing:

No testing could be done since loading the game is not yet implemented. The addition of this tool is not of high priority for the drop-dead (Rev 1 DEMO) date. (**Requirement #4**)

1.5 Load Game

Verification:

The user is currently unable to load the game as the save function is not yet implemented, since these two functions go hand-in-hand once the save function is fully operational the loading function will also be working. We do not meet this requirement since loading does not currently work, we are unsure if this will be implemented.

Validation:

The player must be able to load their game from a saved state in order to resume progress in their game so they do not need to complete the game in one sitting. Our game is lacking this functionality and therefore fails to meet this requirement, it will be implemented for the next version of the game.

Testing:

No testing could be done since loading the game is not yet implemented. The addition of this tool is not of high priority for the drop-dead (Rev 1 DEMO) date. (**Requirement #5**)

1.6 Main Menu

Verification:

The game has a main menu but lacks the features to load a game or adjust graphical and game-related options. The main menu consists of 4 buttons (New Game, Load Game, Options and Exit) however only the new game and exit buttons are functional. The save/load system will be introduced at a later time, but will probably not make the drop-dead date of the rev 1 demo.

Validation:

The user needs to be able to access a main menu in order to select options such as loading and starting a new game as well as graphical options/settings in addition to closing the game from the main screen . The menu has been implemented in the game using Flash to make a Scaleform file with selectable buttons, which is called on a blank map, when the new game button is pressed, the first map in the game loads.

Testing:
Testing was done by loading the initial blank map checking to see if the menu appeared, and then selecting the new game button to load the first map of the game which was completed in a reasonable amount of time (~1-20 seconds). The main menu was also tested to close the game by pressing the exit game button which was completed in a reasonable amount of time. (**Requirement #6**)

1.7 Options Menu

Verification:
The options menu has not yet been implemented. The following is a description of the planned Options menu functionality. When the user pressed the “Escape” key then they are showed a menu which contains several options such as save, load, exit etc. Options is one of these selections, and clicking it brings up another menu with several additional choices. These choices are input configuration, audio and graphics. Input configuration should allow the user override default key-bindings for actions and map them to custom keys. Audio should allow the user to change the volume of music and sound effects. Graphics should allow the user to change certain render settings of the UDK engine.
Validation:
The Options menu has not yet been implemented and as such validation can not be completed at this moment for this feature.
Testing:
No testing could be done since the options menu is not yet implemented. This functionality is not of high priority for the drop-dead date of the rev 1 demo. (**Requirement #7**)

1.8 Stat Menu

Verification:
When the player presses the “c” key then they are presented with a menu which shows a preview of their character’s main statistics in real time.
Validation:
This current design implementation is sufficient for the user’s need to allow for careful adjustment of the player’s play style as well as allowing the player to see the full statistic change of all equipped passive and active related tablets that affect damage, movement speed, attack speed, health, mana, regen, or critical strike.
Testing:
To test this feature, the user presses the “c” key and waits for the statistic menu to appear, pressing the key again results in the menu removing itself from the screen. This test was successful and occurs within a reasonable amount of time (~instantaneous). Equipping a tablet with the menu open updates the statistics in real-time.

1.9 Inventory

Verification:

Currently the inventory screen meets the requirement of holding tablets in the inventory and allowing the user to equip and unequip tablets. It fails to meet the requirement of holding weapons, miscellaneous items and quest related items, this is because those features have not been implemented yet, however extension of the inventory from tablets only to other items is not difficult and only requires that the code for the remaining items be extended on the TS_Items UC script. The player can successfully transfer tablets in the inventory to other inventory slots, or passive slots/active slots without duplication of the items or loss of the items 100% of the time tested. When the inventory screen is removed, the organizational structure of the items laid out in the inventory retain their order when the screen is brought back, 100% of the time tested. The inventory is successfully brought up using the hotkey “i”.

Validation:

The current design implementation is sufficient for the user, allowing the user to successfully manage their tablets in terms of spatial organization as well as clear indications to the player of what tablets are being used for passives, which ones are being used for actives, and which ones are simply being held in the inventory.

Testing:

The user is presented with their inventory when the “i” key is pressed, appearing within a reasonable amount of time (~instantaneous). The inventory menu can be removed by pressing the “i” key again, restoring the HUD. The user can manipulate the configuration of their collected tablets by clicking and dragging them to different parts of the screen. (**Requirement #9**)

1.10 Quests

Verification:

The quest screen has been implemented, the goal is to display to the user a quest screen when they press the “q” key in-game. The quest screen displays all active quests, as well as all completed quests. There is one possible selected active quest, indicated by a special marker next to the quest’s name. This can be changed by double-clicking another quest, or by selecting one and pressing the “Change to current quest” button. This functionality is not implemented yet, and instead the quests are shown on the HUD (max 3) in the order which they were received in (no ability to choose which quests to show). Each quest, either completed or active can be examined to see progress of the quest (as well as the subquests involved i.e. steps). Each completed portion is indicated with a special marker next to it, and the currently active portion is highlighted, with a description written in the description panel. The user can select other steps of the quest to view their descriptions as well. The indication of completed vs. active quests has not been implemented and is low priority, but the progress can still be viewed.

Validation:

This design implementation is sufficient for the player. The user needs to be able to open and close their quest log to keep track of completed quests, as well as quests in progress, which this system provides. The quest log also allows the player to enhance their game lore by reading in-depth quest descriptions. That being said, although the implementation is sufficient it is not optimal as being able to select quests to display and seeing the difference between completed quests and active quests would be useful to the player.

Testing:

To test the design implementation, the user presses the “q” key to bring up the quest log within a reasonable amount of time (~instantaneous) and then the “q” key can be pressed again to remove the quest-log and restore the HUD. The on screen quest progress trackers were tested by receiving 2 quests in a specific order, and appearing on screen in the same order. The progress of each quest was tracked in real-time (tested vs. a kill quest) and each time a target was killed, the quest tracker updated within a reasonable amount of time (~1-2 ms), the quest tracker would disappear immediately from the screen when the quest was completed. (**Requirement #10**)

1.11 Pause Menu

Verification:

The pause menu is opened when the player presses the “esc” key, revealing the options button, save button, load button and quit game button. The actual game itself is also paused, preventing game-update loops from occurring (mechanics are all stopped including AI).

Validation:

The design implementation for the pause menu is sufficient for the player, as it provides access to the save, load and quit game functionality, whilst also pausing the update cycles in the game initially, however it fails to keep this criteria met, as a bug unpauses the game when any mouse input is received.

Testing:

To test this design implementation, the “esc” key is pressed to open the pause menu within a reasonable amount of time (~instantaneous) displaying it on screen, while pausing all updates of the game in-progress (no player movement). An option on the menu is pressed to test the pause functionality however the pause state is broken and continues to update the game. The “esc” key is pressed again to remove the pause menu from the screen. (**Requirement #8**)

2. In-Game Requirements

2.1 Equipping Stat Changing Tablets

Verification:

Currently the “equipping stat changing tablets” event does work, allowing the user to move the tablet in the inventory to a passive slot by dragging the tablet to the slot. It fails to meet the requirement of forcing the user to be in a safe zone to equip the tablet. The necessity of such a function is under re-evaluation. Additionally, the gameplay effects of each tablet on the statistics of the player’s character can be seen to update in real-time (~1 tick) and correctly effect the statistic that the tablet is scripted to change. Switching of tablets reacts in real-time and has been tested for duplication of items, or any other unintentional functionality. One-hundred percent of the time, equipping tablets does not duplicate the tablet. An additional requirement that the statistics should not be duplicated or kept without the appropriate tablet has been tested and 100% of the time, the statistics are updated as expected with no duplication in statistics.

Validation:

The current design implementation of the “equipping stat changing tablets” event is sufficient for the needs of the user, allowing the user to successfully move the tablet from an inventory slot to a passive slot. The UI elements of this feature and backend work to allow the user to customize their character and their play style, additionally the tablets successfully change the appropriate statistic of the user and are reflected clearly in the “statistics” tab of the stat screen providing the user with current information about their pawn’s gameplay related stats.

Testing:

The user can enter the inventory menu and arrange their tablets by clicking and dragging the tablet’s icon to the specified location. Releasing the tablet on a passive slot will equip the tablet to that slot, swapping the contents of that slot with the contents of the inventory slot the tablet was dragged from. This swap occurred within a reasonable amount of time (~1 ms) and no duplication of items or stats occurred 100% of times tested. (**Requirement #11**)

2.2 Equipping Ability Changing Tablets

Verification:

Currently the “equipping ability changing tablets” event does functionally work, allowing the user to move the tablet in the inventory to an active slot by dragging the tablet to the slot. It fails to meet the requirement of forcing the user to be in a safe zone to equip the tablet. The necessity of such a function is under re-evaluation. Additionally, the logic and UI for this function as well as the spells are fully scripted. This function reacts in real-time (~1 tick) and has been tested for duplication of items, or any other unintentional functionality. One-hundred percent of the time, equipping tablets does not duplicate the tablet. The hotkey associated to the active slot with a spell assigned to it, does cast the appropriate spell when pressed.

Validation:

The current design implementation of the “equipping ability changing tablets” event is sufficient for the needs of the user, allowing the user to successfully move the tablet from an inventory slot to an active slot, swapping the contents of both slots with one another, assuming the tablet selected is an active tablet. The ability possessed by the tablet is appropriately assigned to the correct hotkey and allows the player to cast the spell associated, allowing the player to adjust to a unique style of gameplay.

Testing:

The user can enter the inventory menu and arrange their tablets by clicking and dragging the tablet’s icon to the specified location. Releasing the tablet on an active slot will equip the tablet to that slot, swapping the contents of that slot with the contents of the inventory slot the tablet was dragged from. This swap occurred within a reasonable amount of time (~1 ms) and no duplication of items or stats occurred 100% of times tested. When the hotkey assigned to the slot with an active tablet in it was pressed (keys 1-5) the appropriate spell was cast by the player successfully. (**Requirement #12**)

2.3 Talking to NPCs

Verification:

We are currently using the player model as the friendly NPC model in the game, the player does have the ability to interact with the NPCs in order to talk to them, exchange items with them or accept quests from them. The player should be able to initiate interaction with friendly NPCs whenever they choose by approaching the NPC (assuming it can be talked to) and pressing the interact key “e”. This opens up the dialog menu which allows the player to choose from a set of possible responses to an NPC response, depending on the response the player gets a new set of possible responses to choose from. This system can also work with other items for lore and story including books or signs as long as they use the TS_Pawn class as a base.

Validation:

This design implementation is sufficient for the player to progress through the story as well as advancing the plot and initiating quests. The implementation fails to meet the specification criteria of being able to mouse click on the NPC that needs to be interacted with, however the implementation still allows the player to interact with the NPC by pressing “e” as long as they are within a certain radius of the NPC.

Testing:

To test the functionality of this implementation the player approaches an NPC or pawn that is flagged with the bConverse bool set to true in unrealscript. At this point there is no indicator as to who the player can talk to or can’t talk to. When the player is within 10 unreal units of the pawn, and the “e” key is pressed, the conversation menu is opened providing the player with the information “Pawn Name”, the pawn’s response to the player, and a list of possible responses the player can give, this all appears within a reasonable amount of time (~1-10 ms), when the player double-clicks on a response, that response should be fed to the conversation manager and a new NPC response should be returned with a new set of player choices, until the end of the conversation is reached by a specific player response (fed with an NPC response of -1 which closes the conversation). When the conversation is done, the conversation menu is removed and the HUD is restored within a reasonable amount of time (~1-10 ms). This test is repeated multiple times to test the outcomes of many different conversation branches. For the branches tested, the results were all the expected results based upon the design of the conversation tree. (**Requirement #13**)

2.4 Attacking with a Basic Attack

Verification:

The user is able to find a weapon in game and equip it. Currently, all UDK default and custom weapons are able to be used, but their availability is limited to what is placed in each level. Weapons held by the player are lost when a new map is loaded. The player physically holds the weapon in their right hand, swinging it when the left mouse button is pressed. Any enemy that the sword touches while it is in motion will take damage.

Validation:
The weapon logic is currently implemented and working as intended. The user swings the weapon with left click, performing the specified attack animation and dealing the appropriate amount of damage to any enemies that are hit.
Testing:
The user presses the left click button on their mouse while standing at varying distances from an enemy. Damage is inflicted when the sword's model overlaps the enemy's model (**Requirement #14**)

2.5 Using a Targetable Ability

Verification:
Targeted abilities have now been implemented as of revision 1. The user can equip and use a wide range of abilities, with some of them being targetable. Abilities may be equipped and used at any point in the game, and may further be modified with the use of ability changing tablets. Currently, there is no animation for using an ability, but this has not been deemed an important enough asset to include for the first revision of the game, and is low priority, but could still be included with rev 1.
Validation:
The use of numerous targetable abilities allows the player to solve problems by making a larger variety of choices when dealing with enemies in combat. Pressing the mouse is no longer part of the implementation, and therefore this implementation fails to meet the fit criterion however the other aspects of the fit criterion are met, and the click is replaced by pressing the associated hotkey to cast the spell.
Testing:
In the inventory menu, the user drags an ability onto an active slot. The user then exits the inventory menu and presses the key bound to the corresponding active slot to use the ability. For abilities classified as targetable abilities, the ability shoots out from the player and heads in the direction of the user's current pawn rotation (direction character is facing), assuming enough mana is available. (**Requirement #15**)

2.6 Using a Non-Targetable Ability

Verification:
Non-targetable abilities have now been implemented as of revision 1. The user can equip and use a wide range of abilities, with some of them being non-targetable. Abilities may be equipped and used at any point in the game, and may further be modified with the use of ability changing tablets. Currently, there is no animation for using an ability, but this has not been deemed an important enough asset to include for the first revision of the game. The active spells that are non-targettable are cast by assigning the tablet with the active spell to a slot and pressing the associated hotkey (1 through 5) to cast the spell.
Validation:
The use of numerous targetable abilities allows the player to solve problems by making a larger variety of choices when dealing with enemies in combat. The implementation is sufficient for the

player’s uses, allowing the player to cast spells (by pressing associated hotkey) at targets without having to position the pawn in the direction of the target, only requiring the player to be within a certain radius of an enemy (either auto-target or AOE spells, which are both non-targetable because aiming cannot be controlled).

Testing:

In the inventory menu, the user drags an active tablet onto an active slot. The user then exits the inventory menu and presses the key bound to the corresponding active slot to use the ability. For abilities classified as non-targetable abilities, the ability has an effect that is not in any way related to the position of the user’s pawn rotation, only the pawn radius from an enemy (i.e. AOE or auto-target). If the player’s pawn is within the designated radius to hit a target, the spell cast randomly targets an enemy within the area specified, applying the effects of the spell, assuming enough mana is available. (**Requirement #16**)

2.7 Map Menu

Verification:

The map menu has not yet been implemented. It’s implementation is not deemed of very high importance (due to a proposed in game minimap), but will be done if there is time.

Validation:

The map menu has not yet been implemented and as such validation can not be completed at this moment for this feature.

Testing:

No testing was done as the map is not yet implemented. (**Requirement #17**)

2.8 Player Death

Verification:

Currently, when the player dies, the screen simply locks over top of the player while the game continues. The planned player death sequence will be an overlayed screen that appears overtop of the locked screen with the options “Continue” or “Quit”. The “Continue” option will load the players last save (auto-save, or otherwise) and the “Quit” option will exit the player back to the Main Menu.

Validation:

A player death sequence is partially implemented, as the screen locks over top of the dead character and the game continues. The overlayed screen is the only thing left to implement for this feature.

Testing:

The player was killed multiple times and every time the screen proceeded to lock above the player, however the death menu did not appear. (**Requirement #18**)

2.9 Death Menu

Verification:

This feature is going to be implemented as it is deemed of critical status. The planned player death sequence will be an overlayed screen that appears overtop of the locked screen with the options “Continue” or “Quit”. The “Continue” option will load the player at the last respawn point and the “Quit” option will exit the player back to the Main Menu.

Validation:

The Death Menu has not yet been implemented and as such validation cannot be completed at this moment for this feature.

Testing:

Testing could not be performed as the Death Menu was not yet implemented. (**Requirement #19**)

2.10 Picking up Tablets

Verification:

Currently the “picking up tablets” event successfully allows the user controlled pawn to pick-up the tablet when the pawn is within a certain radius of the tablet designated by the UC script attached to the extended Actor function Touch(). This provides the user with an inventory item based upon the item information embedded in the picked-up tablet. This requirement is successfully met 100% of the times an item is picked up by the user.

Validation:

The current design implementation for picking up tablets is sufficient and meets the needs of the user to be able to pick-up items from dropped enemies to potentially increase their character diversity and enhance character progression. The items can be visibly seen by the user within the inventory screen for inspection.

Testing:

Tablets were dropped by enemies on death multiple times, the player picked them up without fail every time, no duplication of items or loss of items occurred on pickup provided enough space was available in the player inventory. (**Requirement #20**)

2.11 Receiving a Quest

Verification:

Receiving a quest is an important mechanic of the game and has been added. The sequence for receiving a quest is an event triggers the quest (either through location or conversation) and the quest is moved from the “AvailableQuests” array and moved to the “ActiveQuests” array, where it is then visible in the quest log as well as the top left corner provided there is space to display the quest (MAX 3 quests in progress displayed), to track progress.

Validation:

The implementation of the quest system is sufficient for the player as it provides the player with the ability to receive quests during a conversation, advancing the storyline in the game.

However, this implementation does not have functionality to receive quests via entering an area/location at the moment, so that fit criterion is not met.

Testing:

To test this implementation, the player talks to a specific “test” NPC who provides a quest depending on the appropriate conversation choice. This quest then appears in the quest log as well as the quest tracker on the HUD. This quest appears after following the correct order of events 100% of the trials attempted. (**Requirement #21**)

2.12 Completing a Quest

Verification:

Completing a quest is also an important mechanic in the game that works in tandem with receiving a quest. Once a quest is received, the progress is tracked in the corner of the screen as well as the quest log, when the quest progress is completed (i.e. number of enemies killed is met) the quest is removed from the HUD progress tracker. The quest itself is then moved by the quest manager from the “ActiveQuests” array to the “CompletedQuests” array.

Validation:

This implementation is sufficient for the player as it allows the player to identify what quests have been completed. However, there is no visual or audio cue that is applied upon completion of a quest other than the info stored in the quest log, so that fit criterion is not met. This will be fixed for the drop-dead date of the rev 1 demo.

Testing:

To test this implementation, after the player receives the quest “Kill Something”, the player completes the quest by killing 2 enemies on the map. Once the 2 enemies are killed, the quest is removed from the HUD progress tracker. We know the test is successful because the progress tracker only stops tracking when the quest has been moved from the “ActiveQuests” array to the “CompletedQuests” array. (**Requirement #22**)

2.13 Enemy Death

Verification:

Currently the enemy death event triggers correctly and removes the actor from the set of living pawns on the level. The death of an enemy is also accompanied by a proper death animation, in this case it is a ragdoll of the pawn. All graphical elements attached to this pawn are removed upon death.

Validation:

The current design implementation of the enemy death event is built upon the structured functions provided by the default UC script extensions.

Testing:

Enemies were killed many times by various weapons and the proper death triggers and animations were played. And all graphical elements that were attached to the pawn were removed upon death (i.e. the floating NPC info bar).

2.14 Triggering a Trap

Verification:

The player is now able to trigger traps which will damage him, certain traps will also instantly kill the player. These traps come in a variety of shapes and sizes, forcing the player to adapt to the situation in order to survive. The traps will currently cause the player the correct amount of damage however some of them, such as darts coming out of the wall, do not disappear after hitting the player. This issue was fixed in the latest version of the game.

Validation:

The player is expecting the paths in the levels to be challenging in some ways and one of the most common way to challenge players is to include traps. The player feels challenged since the traps increase in difficulty as he progresses in the level. The traps meet the needs required by the user since they deal damage upon contact with the player.

Testing:

The player ran through traps multiple times and it was shown that the player takes correct amount of damage and the projectiles disappear upon contact with the player.

2.15 Player Takes Damage

Verification:

The HUD displays damage over time effects currently affecting the player, as well as the damage the player has taken. The particle effects also show what damage over time effects the player is taking. These are all evaluated using the Damage_Type base class within the unrealscript. Depending on the type of damage received (all damage done has a damage type) the appropriate damage over time effects are applied.

Validation:

This is sufficient for the player’s needs in order to provide the player with visual cues, allowing them to notice the current state of the game and their character regarding their health and success (i.e. watching health bar, and effects applied like poison or burning) so that their play style can adapt.

Testing:

To test the functionality of the HUD damage over time icons and particle effect systems, the player equips a spell of each type (poisonball for poison, fireball for fire, etc) and moves the pawn close to a wall. The player then attacks the wall, these spells apply on hit effects in an area of effect explosion. It can be seen that the poisonball successfully applies poison to the player’s character (both HUD icon and particle effect appear), this applies to all other spells with similar properties (i.e. fireball). The player’s health bar also changes, reflecting the current state of the player’s character. Multiple effects can be applied at the same time, successfully demonstrated in the test case above. (**Requirement #23**)

3. Additional Requirements

3.1 Updating NPC Info Bar

Verification:

Currently the updating NPC info bar follows the requirement that it must be small enough to fit on screen, and large enough to be easily read by the player. The info bar successfully removes itself from the screen upon death of the associated pawn. The updating bar does correctly update the stats reflected for the NPC attached to the visual element correctly within real-time (~1 tick).

Validation:

The current design implementation of the NPC Info bar is useful to the user, providing the user with the current health of the enemy in real-time, so that the user can see how much damage he/she is doing and how much health the enemy has left. The current info is limited to health of the NPC however and name of the NPC, this can be expanded easily through modular code and scaleform changes to reflect the level if necessary, at the moment we do not find it a necessary requirement through playtesting.

Testing:

To test this feature, the player needs to be within the vicinity of an enemy or friendly NPC. Upon having an NPC on screen, the NPC Info bar should appear above the NPC. The correct NPC name and health should appear on the info bar, and the health bar should reflect the percentage of health the NPC currently has when compared to their max health. The player should then attack the NPC, and the correct amount of health should be removed from the displayed amount, and the health bar should be altered to reflect that amount along the x-scale.

3.2 Heads-up Display

Verification:

Currently the HUD does meet most of the requirements listed in the design document. The HUD does display the current health of the user, the current mana of the user and the current passives of the user in real time (~1 tick). The HUD does allow the user to open the inventory, the quest list, stats screen and pause menu however it does not allow the user to open the map.

Validation:

The current design implementation meets some of the needs of the user, allowing the user to quickly access the important information about the character: current health, current mana, and current passives and actives as well as status effects. The current design fails to provide the user with a way to access the map menu.

3.3 Tutorial Level

Verification:

The tutorial is included in the game to help the player understand the concepts we introduce to the game since some of them may be new to some players. Every action is singled out in specific sections to force the player to use them and familiarize themselves with them, and by the end of the tutorial the player must use combinations of each ability in order to successfully complete the tutorial level. So far the level is about 70% completed and does not fully comply with the requirements but is coming along well.

Validation:

The player must be able to feel at ease with the abilities and commands of the game as well as familiarize themselves with various concepts. The tutorial level is designed in order to do exactly that and so far it achieves this purpose rather well. The level will soon be completed.

Testing:

The level has been played through by players which did not design the level. It was found that the players learned the concepts of the game well enough and rather quickly.