Software Testing Report

Team name: Endeavour (Team 28)

Team Members: Sarah Berry, Finley Brown, Yupeng Di, William Griffiths, Kurtas Joksas, Fraser Masson

4.a

Unit Tests

We started by implementing unit testing that could be used for our continuous integration. We used Junit 4 with Tom Grills testing skeleton (https://github.com/TomGrill/gdx-testing) to test some of the classes.

Our aim was to implement unit tests that cover as much of the code as possible. However, we encountered many problems when attempting to implement these tests. A source of many of these issues is the lack of modularity in the code; frequently the UI and graphics of the game are handled in the same class as the underlying game logic. For example the gameEntity class, and all classes that extend it, includes a rendered which causes an error to occur when attempting to create an instance of the class for testing purposes. Another key issue preventing unit testing is that in the World class many methods and attributes are, potentially unnecessarily, made static meaning that no tests can be run on this entire class and an instance of this class cannot be created for testing purposes. This is a significant issue as the World class is very key to the workings of the game and is passed into many methods in other classes.

After looking over the code and attempting to perform minor refactorings to it, we discovered that to refactor the code to make all of it suitable for unit testing would require us to rewrite the majority of the game. Therefore we decided to avoid major changes to the code and to instead write unit tests for whatever methods we could. Additionally we tried to make our new features as testable as possible by making it modular and avoiding the World class where possible.

Manual Tests

Manual testing was important for this project as it allowed for us to cover the holes left by our incomplete unit testing. We focused on testing any visual elements, the graphics and UI of the game, and any features that couldn’t be tested using unit testing.

We then tried to do manual tests for the UI and any other things we could not unit test.

4.b

Screenshots of our manual tests can be found on our website
https://github.com/jonWadman/auber2/blob/website/docs/pdfs/Manual%20Test%20Screenshots.pdf

Manual Tests

Test Outline

ID	Description	Input	Related requirements
M.1	Test functionality of teleporter	E (when on pad)	FR_TELEPORTER
M.2	Test whether infiltrators are in the prison after arrested	Left click (on exposed infiltrator)	FR_PRISON
M.3	Test whether buttons lead to a game state of corresponding difficulty	Left click (on difficulty button)	FR_MENU, FR_LEVEL
M.4	Test healing bar	N/A(player in infirmary)	FR_HEAL
M.5	Testing the player can move into free space	W (when no wall above player) A (when no wall to the left) S (when no wall below) D (when no wall to the right)	FR_PLAYER_TILES
M.6	Testing the player cannot move into a wall	W (when wall above player) A (when wall to the left) S (when wall below) D (when wall to the right)	UR_PLAYER
M.7	Testing whether the player is notified of sabotage attempt	None	FR_ATTACK_NOTIF, FR_MINIMAP
M.8	Testing infiltrators destroy system and texture updates	None	FR_KEY_SYSTEMS
M.9	Test if the user returns to a prior saved state	K(for save) L (for load)	FR_SAVE
M.10	Test if the beam is larger when ability is active	Left Click	FR_AUBER_POWERS
M.11	Test each ability that the aliens have e.g. slow etc.	None	FR_HOSTILES_ABILITIES

Testing outcome

ID	Steps to perform test	Expected output	Actual output	Pass/ Fail	Rational
M.1	User walks onto teleport pad User presses E Repeat for all teleport pads	Player teleported to a random teleporter pad	Player appears on different teleport pad from every teleport pad and camera follows	Pass	Unable to unit test teleportation as it is not a stand alone function. It is programmed in Player.update which takes world as a parameter.

M.2	User exposes infiltrator by shooting it with beam User shoots beam again to teleport infiltrator to prison User walks to outside of prison to check if infiltrator is present inside	Infiltrator should be visible in the prison	Infiltrator appears in brig	Pass	Unable to unit test shooting as it is not a stand alone function. It is programmed in Player.update which takes world as a parameter.
M.3	User from the menu moves the mouse to one of the difficulty buttons, clicks on the button. User moves around to check the number of civilians, repeat for all difficulties	The number of civilians increases the harder the difficulty selected	There are visibly more civilians	Pass	Difficulty is set in work which cannot be created in a test environment
M.4	Player allows health to be reduced slightly Player moves to infirmary	Players health bar increases to max health over time	Players health bar increased then stopped	Pass	Related to UI Heal itself is unit tested
M.5	Use corresponding key to walk into free space	Player moves into free space	Player moves into free space	Pass	Related to UI
M.6	Move to wall in test direction Walk into wall Check player is not moving into wall	Player does not move into the wall	Player does not move into the wall	Pass	Related to UI
M.7	Wait till arrow goes red Go to location and check if attacker is infiltrator by shooting	Arrow and text of system on UI turns red	Only arrow UI if window resized	FAIL	Related to UI
M.8	Wait by the system until the infiltrator arrives and attacks. Wait till system texture changes to broken	Red flashing lights on system followed by broken system texture	Red flashing lights on system followed by broken system texture	Pass	Related to UI
M.9	Create a save state, play the game for a bit and then load the save state, see if the game goes back to the state it was in when the player saved	The state should return to the original state	The game returns to the state when saved	Pass	Saves the world so cannot be tested
M.10	Play the game until you gain the beam upgrade, activate power then compare beam widths	The beam's area of effect increases when the ability is active (beam is bigger)	The beam appears bigger and it is easier to shoot infiltrator	Pass	Related to UI
M.11	Move to infiltrators and attack them to reveal them. Allow them to use their powers on the player	Each ability performs its corresponding effect	Player slowed, walks opposite direction and black screen appears	Pass	Related to UI

M.7 failed due to the first team not implementing the HUD correctly. Unfortunately we did not have time to fix this as it is different on different screen sizes, meaning it needs to be tested by many people for it to pass. Some people in our team could see the HUD of systems correctly after starting the game but others had to resize the game and could not play in full screen mode if they wanted this feature to work.

Unit Testing

Refer to
https://github.com/jonWadman/auber2/blob/website/docs/pdfs/Unit%20Testing.pdf

ID	Class	Method	Pass/Fail
1.1	Utils	randomFloatInRange()	Pass
1.2		randomIntInRange()	Pass
1.3		randomListItem()	Pass
2.1	Navigation Mesh	setCell(int x, int y, boolean value)	Pass
2.2		cellAccessible(int x, int y)	Pass
2.3		getWorldCoordinates(int x, int y)	Pass
2.4		getSuccessorNodes(PathNode node, int[] destination)	Pass
2.5		generateTilemapPathToPoint()	Pass
2.6		getFurthestPointFromEntity(GameEntity entity)	Pass
2.7		getEuclidianDistance(float[], float[])	Pass
2.8		getEuclidianDistance(int[], int[])	Pass
3.1	PathNode	toString()	Pass
3.2		equals(object)	Pass
4.1	GameEntit y	getCenterX()	Pass
4.2		getCenterY()	Pass
4.3		getCenter	Pass
5.1	Player	powerStopInfiltratorPower()	Pass
5.2		powerOn()	Pass
5.3		respawn(float x float y)	Pass
5.4		heal(float rate)	Pass

C.
Unit testing can be found on our website
https://github.com/jonWadman/auber2/blob/website/docs/pdfs/Unit%20Testing.pdf
Screenshots of our manual tests can be found on our website
https://github.com/jonWadman/auber2/blob/website/docs/pdfs/Manual%20Test%20Screenshots.pdf
Our traceability matrix can be found on our website
https://github.com/jonWadman/auber2/blob/website/docs/pdfs/TraceabilityMatrix.pdf