**GROUP PROJECT, GROUP 3**

**DATE: 29 November 2018**

**TIME: 14:30 – 15:15**

**ATTENDEES** Tom Gibbs, Henry Crofts | Michael Warburton (Unity Master Trainer)

**LOCATION:** IWIC Seminar Room

**Minute Taker: Tom Gibbs**

**Overall aims of the current sprint *(Detailed tasks, user stories and time allocations are tracked on JIRA)***

* Continue to add as much functionality to project build as possible to enable playtesting as early in the project timeline as possible
* Work towards solving Git merge conflict issues and prevent repeats in future

**Meeting:**

Michael Warburton (Unity master trainer) gave a talk and mini workshop to students. The team stayed behind following the seminar to show our project to Michael for his feedback.

Michael asked for a quick elevator pitch before being shown the game, which he timed.

Team gave summary (which took 16 seconds). Michael said the team successfully conveyed the key design elements – he understood what the game was and how it would appeal to players.

He said the only element missing from a brief pitch was the re-sellable element of the product.

“an elevator pitch should contain the games hook (addiction) and the resalable elements (prolonged profit), as to investors it is all about money and its ability to return on their investment”.

He said team introduced the product appropriately: in a hard a cold manner, avoiding description of visual elements and backstory and speaking about the mechanic – “you are pitching a game, not a video, not a book”.

Michael was shown and was able to interact with the mechanics currently implemented.

Michael understood the aim of the player and said if the team work the project to a complete and polished experience, there is no reason it should fail.

Michael then advised that his roles before his current ‘master trainer’ had focused on project optimisation and that as no other students had stayed to discuss projects he would be happy to look through the team’s project and run a quick workshop on how Unity projects can be optimised.

Michael started by using Unity’s profiler and frame renderer monitoring tools to identify the most demanding elements of CPU/GPU resource.

Went on to detail that monitoring process times of GPU/CPU events are the most descriptive indicators of project efficiency (more so than frame rate and batch rate).

Explained that multiple draw calls were using much of the machines resources – found that dynamic batching was not enabled in project settings. This eliminated a significant number of calls.

Explained that static batching can also be defined to reduced draw calls even further. This can be done so long as each batch of models share the same material and have a sum vertex count of less than 300.

Michaels advised that this batching process is highly efficient but is best avoided with more complex models (highly irregular model angles) even if less than 300 verts as this can produce artefacts in gameplay.

Michael then looked at the quality settings of the project.

Amending screen draw distance.

Amending Anti-Aliasing settings.

Collision detection of all rigidbodies.

Use of text mesh pro for UI elements as it is more optimised.

Advising to use object pooling in place of instantiate and destroy methods.

Michael then went on to explain that if UI elements are interactable, 2 sub-canvases should be used: a dynamic canvas (more interactable elements/elements that change) and a static canvas (for elements that never change/interact). This will prevent unnecessary draw calls to static elements and only update the dynamic canvas (as animations/draw calls are called in order from the bottom of the list until the ‘canvas’ is reached).

Michael also showed the team that a raycast event is present on all UI images included in a project. Unchecking this option will stop these elements polling for user input. This should also be done to all elements in a static canvas to increase performance.

Using TextMeshPro also saves CPU power for text elements. It creates materials and creates sprite sheet for UI elements. These materials can then be batched just like object material.

The ‘heat map’ letting overdraws be shown in the editor was also shown. This visualises which elements are drawn and how frequently they are drawn, showing where to reduce calls to increase performance.

Michael also advised team to consider use of “mipmap” to show size and quality of objects if over rendering (too much detail) or not enough detail, and UV overlaps.

For our project design, Michael recommended use of pro-builder (package manager -> all -> pro-builder). This is an in-engine 3D object creation tool. The complexity of models that can be created, and ability to unwrap UVs in-engine match the quality of the teams asset requirements.

Michael also advised that functions of scripts can be tested in Unity’s profiler.

Must be included:

Using UnityEngine.profiling;

Then surround section of code with:

Profiler.beginSample(“stringName”);

//code to be analysed

Profiler.endSample;

Team will meet after the next group project session, Monday 3 December, to discuss feedback from tutors.

***Detailed tasks, task descriptions, user stories and time allocations are tracked on JIRA.***

**Tasks for the current week:**

**Tom (12 Hours 30mins):**

* **Create 'treasure island' scene (1h)**

Set up treasure island scene to play if team win condition is met. Thought must be given to layout of scene to allow to camera movement during victory cutscene.

* **Create 'chest' animated opening and particle effects (30m)**

Within ‘treasure island’ scene, create animation to open the chest’s hinged lid. Particle system must be created, with the aim of leaving players with a momentary feeling of suspense and anticipation.

* **Create cannon-fire particle effects (20m)**

Create particle effects to be played on player’s cannon fire to emphasize player actions.

* **Create damage particle effect (20m)**

Create particle effects to be played on cannon ball impact, to emphasize wood damage suffered.

* **Create D-pad UI selection script (2h)**

Create script to show D-Pad selectable UI panel over the hold on the ship deck. Players should be able to press the corresponding direction to retrieve items. Items will have a recharge cooldown once selected.

* **Create player UI timer and ID system (2h)**

Create script to display coloured ID circle beneath each player. UI circle will function as a ‘clock-face’ to telegraph current task duration.

* **Create D-pad UI recharge scripts (1h)**

Create a script to impose a cool down period on any item that is selected from the hold, before any players may retrieve it again.

* **Include UI icons within D-pad UI (20m)**

Include UI icons within unity project (amend images if necessary).

* **Update 'Whale script' to play animations, particle effects and throw from boat (2h)**

Update ‘Whale’ script to interrupt boat tasks, play animation and particle effects. Animation must be improved from current implementation. Particle effect to be created as part of task.

* **Create 'Bucket' script to handle bailing of water (1h)**

Create script to allow player to pick up bucket, move with bucket, drop bucket or bail water with bucket.

* **Create buoyancy script to effect on-deck interactables (1h)**

Create script to give interactable items on the ship deck buoyant behavior when the water level is raised.

* **Update 'seagull' script, fixing current bugs and undesirable behavior (1h 30m)**

Edit and reimport model to allow for accurate pivot center. Improve spawning behavior so that all spawned intersect above boat center. Adapt current spawn functionality to a true random position on circular perimeter around ship rather than spawn points.

**Henry (12 Hours 15 mins):**

* **Create main menu screen (1h)**

Create a new scene containing selectable options to transition to the games other scenes. Produce as per planned layout discussed in studio-jam.

* **Create script to handle transitions between scenes (1h)**

Create script to handle scene transitions. Create overlay panel and associated behaviour to allow for fade to black in between each scene.

* **Update 'Mop' script to provide cleaning function (1h)**

Update ‘Mop’ script so mop can be used to erase seagull poo prefabs from ship deck.

* **Update 'Torch' script to allow cannonball to be fired at enemy (1h)**

Update ‘Torch’ script to allow player ship cannons to fire if appropriately loaded.

* **Update 'Enemy' script to allow enemy movement, cannon animation and destruction (2h)**

Update scripts to allow for random spawn at either spawn position, movement to opposite end of screen and player cannon telegraphing when the enemy is in range.

* **Update 'Enemy Cannonball' script to allow hit placement, firing from enemy ship (3h 15m)**

Create script to handle enemy cannon fire if conditions are met. Cannon ball needs to select appropriate space to land on ship deck, telegraph this to player using designed UI overlay and move cannonball along believable path to reach impact point.

* **Update 'Enemy Cannonball' script to damage ship and existing damage within a radius of effect (2h)**

Update ‘Cannonball’ script to effect damage on contact with either ship. Cannonball should include radius check to determine whether existing repaired damage is ‘reactivated’.

* **Create 'Wood' script to allow for deck damage to be repaired (1h)**

Create script to allow selection from hold UI menu, carrying/dropping/use by player to repair a damaged area.