Table of Contents

Table of Contents 0

Skyrates 2

- Development Environment 2
  - Unity3D 2
  - Visual Studio 2
  - Git 3
    - File Structure 3
    - Repository Information 3
  - Photoshop 6
  - Redmine/Pineapple 6
  - Skype 6
  - Discord 6
  - Maya and 3DSMax 6
- Delivery Platform 6
- Risk Analysis 7
- Analysis Format 7
- [Eng] Controls / Controller Scheme 7
- [Eng] Movement 8
- [Eng] Basic Ship Stats 8
- [Eng] Basic Ship AI 8
- [Eng] Ship Combat 8
- [Art] Ship Combat 9
- [Des] Level design (open world) 9
- [Art] Environment 9
- [Eng] Ship Customization 10
- [Art] Ship Customization 10
- [Des] Balancing Ship Stats 10
- [Eng] Advanced Ship AI 10
- [Des] Multiplayer Co-Op 11
- [Eng] Multiplayer Co-Op 11
- [Eng] Humanoid Characters 11
- Art Pipeline 12

Concept to Digital 12
Digital to Repository 12
Repository to Unity 12
Unity to Game 12
Design Pipeline 13
Unity Engine 13
Milestone Updates 14
Milestone 1A 14
Milestone 1B 15
Milestone 1C 15
Milestone 1D 16

Skyrates

Development Environment

Unity3D

Risk: Medium-Low

We are using Unity's 3D game engine to build our game on. In doing so, we are able to jump right into the game's development cycle without having to worry about building the engine itself. Unity does come with some inherent setbacks, such as being unable to modifier the main game loop, but these setbacks are inconsequential for the scale of our project. Unity is a good fit for the team as all have had experience working with Unity prior. It comes with preprogrammed physics, colliders, and an animation machine for a 2-dimenional environment. The programmer(s) are familiar with C#, and has proficient to extensive experience with C++ and other C languages. Unity also comes with in-depth documentation on how to use the engine.

Among Unity's abilities is the ability to make prefabrications (prefabs) which allow the designer(s)/programmer(s) to easily make one instance of an object or unit, and duplicate it while maintaining the same settings between instances. Unity also provides access to public variables in custom scripts so that non-programming team members can test various settings for balancing the game. Unity also comes with a built-in UI system, allowing designer(s)/programmer(s) to dive right into Heads-Up-Display elements or menu screens without needing the programmer(s) to create a structure for it. When looking at publishing options, Unity provides multiple build settings to allow for publishing to multiple platforms, as well as any compatibility that needs to be built into a game to do so. One of the only drawbacks is that the artists have limited experience with Unity, however the programmer(s) are working with them so that the pipeline works with the team's abilities.

Visual Studio

Risk: Low

Visual Studio is being used instead of Unity's built-in MonoDevelop. It is a standard for C-style programming languages and is a very useful debugging platform which Unity has full integration with. The programmer(s) are more familiar and at ease using Visual Studio than MonoDevelop.

Git

Risk: Medium

Git is a version control software which allows the team to host project files on a remote site (the cloud). It will help the team to organize files in a way which all team members can access. While the team is being introduced to Git, at least one programmer has extensive experience in the software and is confident he can support the team as their guide.

All project and document files will be stored long term on Champlain's Pineapple site (built on Redmine). In order to do so, team members will use SourceTree or GitKraken, which are a windows/mas. These clients' purpose is to aid users in the pulling/pushing of content to/from a remote server without use of the command line. In our case, it will be utilized so that team members can pull/push documents and project content between their local machines and Champlain's Pineapple server. This will make using Git easier as it allows team members who are not as familiar with the inner workings of Git to use a graphical interface instead of command line, which can be confusing when first introduced to Git.

File Structure

General Naming Schemes:

- Docs
  - [DOCUMENT] _Milestone[X].[ext]
  - i.e. TechnicalDocument _Milestone1A.pdf

File Structure

- docs – holds any documents for safekeeping in the cloud
  - art - artist docs
  - designing – designer docs
  - production – producer docs
  - programming – programmer docs
- project – holds all project content (this is the Unity directory)

Repository Information

Master (Root/Trunk)

The master branch – under most circumstances – should not be committed to. This is the main source of information for the game. Builds must be built from this branch.

Exceptions: You can commit directly to the master under these exceptions

- Fixing a bug < 1 line of code (see Branches)
- Tweaking 1-2 variables in the unity inspector
- Documents

Branches

Types:

- Features
- Bugs

Naming Scheme: [TYPE]/[FEATURE_NAME]

Examples:

- feature/unit_placement
- hotfix/units_not_killed
- release/v1.0.0

When to use:

All features must start in feature branches. All feature branches should have an associated feature issue in pineapple of the same name.

Large bugs (i.e. >1 line of code) must also be put into feature branches, but do not need an associated feature issue. Rather, these should be individual tasks under one Bug feature issue.

Bugs which do not meet this description can be directly pushed to the master branch.

Exceptions:

When there is only one person active in the repository (i.e. during setup) branches can be ignored. This should only persist for 1 week at maximum.

How to Use:

1. Create feature branch
2. Make changes
3. Commit to branch
4. Test locally
5. Check for merge issues with master branch
   a. Fix merge issues
6. Merge with master
7. Delete feature branch
8. Tag master (see Tags)

Tags

What are tags: Tags are a way to view changes in the repository without digging through the changelog

When to use: When a feature branch is merged with master, when a bug is fixed, when a release is built.

Naming Scheme:

- *Bugs*: X.X._
    - Find last X.X.X tag, increment 3rd digit
    - Examples:
        - 5.1.8 -> 5.1.9
        - 0.0.9 -> 0.0.10
        - 41.542.2152 -> 41.542.2153
- *Features*: X._.X
    - Find last X.X.X tag, increment the 2nd digit by 1, set 3rd digit to 0
    - Examples:
        - 2.4.1 -> 2.5.0,
        - 1.9.0 -> 1.10.0
        - 8.25.8432 -> 8.26.0
- *Builds*: _.X.X
    - Find last X.X.X tag, increment 1st digit, set 2nd and 3rd digit to 0
    - Only done by programmers or designers
    - Examples:
        - 7.1.0 -> 8.0.0
        - 0.63.52 -> 1.0.0
        - 12.5432.534263 -> 13.0.0
- *Milestones*: Milestone_
    - Find last MilestoneX tag, and increment the digit X
    - Should ONLY be done by the producer
    - Examples:
        - Milestone1 -> Milestone2
- *QA*: Milestone_-QA_
    - Find last QAX tag, and increment the digit X
    - Keep the milestone digit from the last milestone tag
    - Examples:
        - QA1 -> QA2

Photoshop

Risk: Low

Both the artist and designer have experience with Photoshop. Photoshop is one of the artist's preferred methods of 2D asset creation.

Redmine/Pineapple

Risk: Low

Champlain’s Pineapple site is run on Redmine, and hosts all Milestone documents, the Git repository, and various information about the project in the wiki.

Skype

Risk: Low

Skype is a text, voice, and video chat software which all team members are familiar with. This will be used to include members of the team who cannot be physically present at a team meeting.

Discord

Risk: Low

Discord is a simplistic and easy to access text chat client which all team members will be using to communicate in an efficient manner. It is hosted by the company, and therefore does not need to be self-hosted.

Maya and 3DSMax

Risk: Low

Artists know one or both platforms very well. These tools are used to create 3D models and art, and are self-sufficient programs. Artists can easily use these tools to export 3D models for usage in Unity.

Delivery Platform

Risk: Low

We are building the game for the Windows PC platform. Unity natively exports to the Windows platform, and it is the easy to manage and maintain. We will be natively including support for Xbox controllers and keyboard/mouse controls. Since the game is multiplayer, we will be adding support for both dedicated servers as well as peer-to-peer play. Dedicated servers will allow for a separate computer to handle communication between players, whereas peer-to-peer structures will allow for players to easily spin up new games in which they can play together.

Risk Analysis

Current team: 1 Designer, 1 Programmer, 2 Artists, 1 Producer
Disciplines to bring on (prioritized): 1 Designer, 1 Artist, 1 Programmer, and 1 Artist

Analysis Format

[Discipline] Mechanic/Feature *Risk with current team - Risk with added developers*

Description of features
Reasoning of risk with current team
Reasoning of risk with desired developers

Included in Prototype

[Des] Controls / Controller Scheme

High – High
Making controls intuitive and efficient while still maintaining good movement in world space is imperative to getting this game to sell well through QA. Find the right balance of a cross between boat, plane, and blimp movement is key to getting this right.
Creating and testing controls schemes have proved difficult. This task remains high risk, even now that the team has more perspective on why we are having difficulties. Striking this balance takes a lot of QA iteration and time from programmers to give designers lots of options.
The risk will remain the same, even with another designer. This requires a lot of iteration. That said, with another designer, being able to delegate time and testing on this will be easier.

[Eng] Controls / Controller Scheme

Medium-Low - Low
Controls are very critical to the game feeling good. They need to be intuitive, and the effect on the game (movement and interactions) need to be balanced nearly perfectly. This also includes kinetic user feedback (controller rumble), which helps with immersion.
Iterating on the controls themselves are relatively low risk due to how controls are set up in engine. Most of this risk lies in the engineering of movement in response to controls. Once the programmer(s) can find the proper resources, implementing haptic feedback will be low risk.
If more programmers are brought on, the ability to iterate on this will increase, making this risk negligible.

[Eng] Movement

Medium - Low

As players provide input to the game, it is expected that their characters move in a way which is expected and intuitive. This movement depends on what kind of control scheme we use. This will need to be adapted as controls change, and thus will require iteration early and often.

This is mildly risky for the current team. Given there is only one programmer, this task will need to be balanced against other tasks and efficiently implemented. This also requires thorough communication with design.

This becomes less risky if we can add another programmer, but more risky if we take on only other disciplines. If we can bring on another programmer of any specialty, then one can dedicate more time to this task. If no programmers are brought on board, then the sole programmer is spread more thinly.

[Eng] Basic Ship Stats

Medium-Low - Low

Basic ship stats include health (amount of damage able to be taken), amount of damage dealt, how fast ships are, and the loot which will be dropped. These stats help make the world its inhabitants fell more realistic and diverse.

This is a low risk item, and also low priority for mid-mortem prototypes. Health and loot have been functionally implemented, but aside from the base systems to include modifiers, the basic inclusion of statistical variables in gameplay is not a high in risk.

With another programmer, delegation becomes easier and this task can receive more attention, and be programmed in a way which is documented well.

[Eng] Basic Ship AI

Medium - Low

Artificial Intelligence is the backbone of any and every form of non-player controlled movement and action. Creating a basis for this was not difficult, and implementing basic move and shoot mechanics was easy.

Moving forward without more resources means this task cannot receive too much attention. Given that it is basic AI, however, means this risk comes in around medium for 1 programmer.

For more resources, this risk goes down, as task delegation becomes easier. AI, in general, can receive more attention, making basic AI low risk.

[Eng] Ship Combat

Medium-High - Medium-Low

In-game ship combat includes interactions such has movement, ramming, and various forms of artillery combat. Movement has already been covered as its own risk.

However, forms of movement such as boosts and slowing down will also need to be iterated upon. Ramming will require dealing damage to the target (which is implemented), and dealing a small amount of damage to the user if the target is not destroyed. Cannons require multiple forms of input for varying types of artillery; broadside, short range, long range.

Given that artillery requires projectile physics, this feature is a higher risk, but not the worst. There is plenty of documentation on trajectory physics, and can be done and iterated upon. With one programmer, only moderately high risk, as it will take some dedicated time.

With more team members, and a lot of testing, this task's risk becomes less. This is because more time can be dedicated to it by one person - especially if we have a programmer who knows gameplay scripts really well.

[Art] Ship Combat

Medium-High - Low

Art that is a part of ship combat includes mostly player feedback. This includes VFX for; exploding ships, loot collection, and effects for projectiles and destruction of ships. The also encompasses the animations of ships/components and models/textures for ships and artillery.

Texturing and materials for these items have a medium low risk, they just take a time commitment. Particles are relatively new to the team, so they have a bit higher risk. For one artist dedicated to effects and environment type art, this risk is average but closing in on new territory.

If we had another artist or two, this task drops dramatically in risk. Having more artists allows for the workload and their various types to be better dispersed and taken on by those who specialize in the area. This task would become more low risk.

Features for Production

[Des] Level design (open world)

Low - Low

The design of the world/level that the player enters into is pretty simple, provided art assets are in and accurate. It requires lots of drag/drop into level and iterative testing by QA. It is low risk for any number of designers, it just takes some iteration.

[Art] Environment

Medium-High - Medium

General environment art is important to give the player a sense of what the environment is like. Art is important. Going further into production, larger art pieces are preferred, so that design can expand into larger encounters and end state.

For our current team standings, this is a high risk. It requires lots of modeling and texturing to make the world feel diverse. Pipeline wise, this task is requires designer world building, and is required for the final design by designers to get the world to feel real.
Moving forward, risk goes down, especially if we can dedicate resources to concept development. This is key for large art creations, such as an end state flotilla.

[Eng] Ship Customization

High - Medium-High
For now, customization is limited to a single set of components with no stats. Going forward, however, customization will be player controlled, you will be able to collect loot, and customize your ship based on what loot you have collected - and have those components affect your overall ship stats.
For programming, these tasks on their own are medium-low risk. Putting them together, however, and including the systems to which let ship customization be functional instead of aesthetic, elevates this risk to high.
With more resources, this risk goes down a little (due to task delegation), but is still relatively high due to the amount of system work this will take.

[Art] Ship Customization

Medium - Medium-Low
Time doesn't change, requires onboarding for more art
Ship components are custom designed by artists to fit a ship in a modular capacity. This takes a lot of time and pre-planning. This means it takes a lot of time and resources. Presently, we have one artist dedicated exclusively to this (and some concept art). This is their largest time sink, but it's getting done, so the risk isn't the worst.
With more resources, we'd like to dedicate modelling of new components to an artist so we can churn those out to design work. This would free up art resources in order for us to work on more stuff.

[Des] Balancing Ship Stats

Medium-Low
This refers to the balancing of ship stats against one another, player v non-player entities. This is a low risk task. It is easy to do, but takes time and a lot of testing.

[Eng] Advanced Ship AI

Beyond Scope - Medium/Medium-High
Moving forward, AI in the game would need to be more advanced to successfully convince the player that the world is active. This means AI must be smart and emergent. It must seem like the entities are alive and make conscious and forward thinking decisions.

However, moving forward, it is imperative this be done well so the world the player explores feels lived in and alive when they are not around. This means AI must be implemented intentionally and efficiently. This cannot be done with one programmer. They would be spread too thin and thus progressing past basic AI becomes out of scope. Adding another programmer to the team, especially one with experience and that derives enjoyment from creating AI, will bring this task back into scope. If the sum experience of programmers on the team is enough to make fairly reasonable AI, the the risk is normalized. Otherwise, if there is not a lot of experience in AI from the programmers as a whole, then this risk remains high, but doable.

[Des] Multiplayer Co-Op

Medium-High

Testing multiplayer interactions is difficult to do. It requires a ton of testing and watching and trial by fire. This item inflates the risk of the actual engineering of the system, as it will help with bugs and takes more time and communication - regardless of resources.

[Eng] Multiplayer Co-Op

High - Medium-High

One of the main goals of the game was/is to allow users to play cooperatively in an open-world on different computers. In order to do so, there must be a network interface to allow for multiplayer-over-internet connections.

For one programmer, this task is almost high risk. Many of the systems are already in place, but bug testing new features through a network interface takes a lot of time and forethought.

This risk will not change with more people when evaluated on its own. It will still take the same amount of time and dedication. Relative to other risks, it will be easier because one person could take over handling network interfacing.

[Eng] Humanoid Characters

Out of scope - High x2

Adding the systems and mechanics to control, interact using, and add combat form, is extremely high risk, bordering out of scope. This will require a near overhaul of mechanics and re-evaluation of current interactions. This is something that we would like to do, but is not possible without a lot more resources.

Art Pipeline

Concept to Digital

1. Discuss / Brainstorm with group
2. Artist and Programmer discuss implementation
3. Artist sketches either a physical or digital illustration
4. Artist gets the go ahead from Designer (if not, go back a step)
5. Asset made in either Photoshop or Illustrator

Digital to Repository

1. Asset is saved
    a. PNG Format
    b. Resolution of 72 pixels per inch and flexible pixels per unit
    c. Files will be organized by a folder structure and named appropriately
        i. Formatting: “Assets/Resources/Textures/*type*/*asset*.png”
        ii. i.e. “Assets/Resources/ Textures /Background/ground.png”
2. Asset is copied to Repository folder under "./project/Assets/Resources/", and to the appropriate subfolder
3. Asset is stashed, committed, and pushed to remote repository to any appropriate branch that is not the master

Repository to Unity

1. Unity will auto import any assets at the specified path when opened
2. Artist or Designer opens sprite/asset in Unity inspector
    a. Modifications made to pixels per unit or slicing made as needed

Unity to Game

(Mostly handled in the Design pipeline)

1. Drag into world space OR make prefab
2. Pixels per unit, various materials, and sprite slicing are all handled via the unity inspector

Design Pipeline

Unity Engine

- **Assets.** All assets can be found under Unity's asset inspector
  - **Materials**: materials
  - **Prefabs**: all Unity prefab objects
  - **Resources**: all art assets
  - **Scenes**: Unity scenes
  - **Scripts**: custom scripts
- **Prefabs.** Prefabs should be made when GameObjects will be reused - specifically for units and spaces. These can be made by dragging an object into the prefabs folder in the asset inspector
- **Sprites.** Sprites can be dragged in from the asset inspector to the world window to be created as a GameObject
- **Colliders.** All interactable/collidable prefabs need a collider to interact with one another (and interact with player selection). Either consult the Unity manuals or a programmer.
- **Scripts.** Programmers will write custom scripts as necessary. These can be attached to their GameObjects by a) dragging the script on top of the GameObject or prefab or b) Adding a script component inside the GameObject Unity inspector (see Unity manuals).
  - **Public Variables**. Public variables will be utilized as much as possible to give other team members the ability to tweak values for gameplay and balancing. Examples of this include movement distance, attack range, team color, and if a unit will overtake other units on attack. Values can be changed in the Unity inspector when a GameObject with the script is selected. To add more variables, contact a programmer to discuss how it can be done.

Milestone Updates

Milestone 1A

- 01/23/2018
  - Jennifer Carlin:
    - I did concept art.
    - I made logos!
  - Ellie Peak: I also did concept art
  - Dustin Yost:
    - I did some preprototyping, made the agenda for our sprint planning, and ran our scrum meeting
    - I updated the presentation (revamped it, added notes for stuff to be filled in) and went on a creative rampage cause it was fun and added possible features to the Skyrates concept.
  - Mish Shea: I wrote down some base mechanics and started sketching up some visuals.
  - Cody Douglas: I asked the team to remember to find time to sleep.
- 01/24/2018
  - Jennifer Carlin:
    - I sketched a basic character concept for Skyrates
    - I organized issues in pineapple and made sure all of mine had due dates and estimated hours
  - Ellie Peak: I sketched a ship concept for skyrates and general concept art for skyrates
  - Mish Shea: I sketched up some mechanics visuals for Skyrates.
  - Dustin Yost:
    - Skyrates: Added vertical movement to player ship, added rotating player (separate of camera), add other ships
    - Attended meeting, Managed tasks for the team, Categorized and fleshed out mechanics according to goals/risks, Edited slides
- 01/25/2018
  - Ellie Peak: Finished the poster art, drew context doodles, and added pictures to the slideshow of skyrates
  - Mish Shea: Finished up the tiles for WW physical prototype. Organized the Skyrates Systems and Mechanics. Diagrammed more mechanics.
  - Dustin Yost: Made a lot of work on the Skyrates prototype, Added mechanics to TechDoc(s), Added content to presentation, tweaked tasks in pineapple, Added tech doc content to slides, Added loots to Skyrates
  - Jennifer Carlin: worked on god concepts for weather wars
- 01/26/2018
  - Jennifer Carlin: I finished the god concepts
- 01/28/2018:
  - Jennifer Carlin: made an island blockout, cleaned up pineapple tasks, and rehearsed pitch
  - Mish Shea: I attended the meeting, I finished the Weather Wars Prototype, and I did the game sketches.

  - Dustin Yost: Attended sprint wrap up meeting, I added Weather Wars prototype to slides
  - Ellie Peak: Attended meeting today, made modular ship demo and added it to the slides

Milestone 1B

- 1/29/2018
  - Dustin: Attended meeting
  - Jensie: attended meeting, worked on skybox and background cloud planes
- 1/30/2018
  - Dustin: brainstormed architecture and tools for programming tasks
  - Ellie: Watched some unity tutorials, worked on the logo, had a meeting with jen about art
  - Jensie: attended art concept meeting and worked on model for Merchant ship
- 1/31/2018
  - Dustin: planned architecture a bit more, attended meeting
  - Jensie: Finished modeling and UVing merchant ship, attended meeting
- 2/1/2018
  - Jensie: made enemy navy ship
  - Ellie: worked with dustin on unity things and began the modular ship
- 2/2/2018
  - Dustin: started networking framework and handshake + Started ship component asset storage
    - Update movement via controller
    - Setup network architecture and components
    - Pull utilities from Networking final and remove excess code
    - Setup packet/event handling
    - Attempt to setup handshake
    - Setup architecture for component tracking and add tools for design
    - Setup Client/Server handshake, Started gamestate updates
    - FINALLY got networking to transfer gamestate
  - Ellie: continued working on the modular ship and parts
  - Jensie: got a scene set up in unity for art and made a material for changing the color of the sails
- 2/3/2018
  - Dustin: Finished adding basic multiplayer support
  - Jensie: learned unity stuff, got skysphere working in art scene. modeled, uvmapped, and exported merchant schooner
- 2/4/2018
  - Dustin: Finished final touches on networking, met with Mish for design stuff, and Ellie for tweaking model pivots. Attended last meeting.
  - Jensie: Attended meeting

Milestone 1C

- 2/5/2018
  - Jensie: went to meeting, made and pushed cannonball and reticle

  - Ellie: went to meeting, made and pushed cannonball and reticle
  - Dustin: went to meeting, made and pushed cannonball and reticle
- 2/6/2018
  - Jensie: went to meeting, made and pushed cannonball and reticle
  - Dustin: Finished most of the restructure, started bug hunting, prepared for incoming gameplay tasks | Decided that I want to make an Engineer's Creed
- 2/7/2018
  - Jensie: nothing
  - Dustin: Fix bugs in entity networking, Move input to AI pipeline, Fix entity serialization and ownership, Add player position updating to server
- 2/8/2018
  - Jensie: textured the crate, will be doing more tonight | updated models, re-exported textures and materials, making an island and TREE!
  - Dustin: Did a lot of networking today, including trying to get non player entities to sync | Updated cannons to fire in bursts and projectiles to move towards player's aim | Updated cannons to fire in bursts and projectiles to move towards player's aim
- 2/9/2018
  - Jensie: made skylands, crate, and crate texture
  - Ellie: worked on the player ship
  - Dustin: fixed one big network bug
- 2/10/2018
  - Jensie: made medium navy ship, implemented art in engine
  - Ellie: i finished modeling the player ship
  - Dustin: ran interference for design work, added audio management system
- 2/11/2018
  - Jensie: made skyland materials ans attended meeting
  - Dustin: Added some 3rd week polish, bug fixed networking and QA, ran interference for art and design during meeting, added basic AI, recorded gameplay
  - Mish: made QA results sheet and analyzed results. Remade slides for the presentation. Made the draft of the GDD. Found sounds. Attended meeting.

Milestone 1D

- 2/12/2018
  - Jensie: Attended meeting, screamed a bit
  - Ellie: Attended art bible meeting, went to pre-milestone meeting, finished modeling and help-implementing the player ship, made some heckin clouds, edited our logo | Attended meeting, UV ship, began rig
  - Dustin: Attended meeting, ran QA with zynab, started systems work for diagetic stuff
  - Zynab: Attended meeting, ran QA (woo), continuing to find my place as a producer
- 2/13/2018
  - Jensie: oh! I updated materials, and made a parachuteless crate, and new materials, and cleaned up stuff in unity also attended meeting today
  - Ellie: I havent worked ok skyrates yet today

- Dustin: Attended impromptu meeting, Added VFX dispatcher for particles, added systems for ship health feedback (more particles), Added controller input for menu | Talked with knowledgeable seniors about analytics options and controller iteration
- Mish: Attended meeting, got test plan done, made the QA test form
- Zynab:
    - Made a twitter
    - templated social media plan
    - Signed up for QA on Sat 2/17
    - started to conceptualize midmortem presentation reqs

- 2/14/2018
    - Jensie: made some particles! Attended meeting
    - Ellie: Attended meeting
    - Dustin: Went to update meeting and continued work on controls | Added base code for getting projectile arcs into game UX
    - Mish: Attended meeting
    - Zynab: went to update meeting, wrote notes
- 2/15/2018
    - Jensie: made particles!
    - Ellie: Worked on rigging basically everything
    - Dustin: added all the risk analysis to tech doc
    - Zynab: I made 2/3 sheets for shirts
- 2/16/2018
    - Jensie: Ahhhh updated particles! started some UI sprites | made sprite for canon in Ui
    - Ellie: Finished rigging, began animations, as seen onthe art page
    - Dustin: Tweaked particles with Jenn, added loot feedback systems
    - Zynab: worked on pres, worked on shirts... am in search of a laser printer on campus that I can direct print to in order to finish.
- 2/17/2018
    - Jensie: did a little bit of UI
    - Ellie: Ran into a lot of issues rigging today
    - Dustin: did nothing, took a break
    - Mish: Went to QA, added so many loot pivots, world building, and updated loot tables
    - Zynab: Launched twitter, helped a lil w/ QA
- 2/18/2018
    - Jensie:
    - Ellie:
    - Dustin:
    - Mish:
    - Zynab:
- 2/19/2018
    - Jensie:
    - Ellie:
    - Dustin:
    - Mish:

  - Zynab:
- 2/20/2018
  - Jensie:
  - Ellie:
  - Dustin:
  - Mish:
  - Zynab: