Photon Multiplayer Minigame Project

Team Systematic Declaration

****Attention Dr.Dan****

****See end of document for extension contributions****

Team Lead	Lead Programmer	Lead Tester	Artist
Bobby	Akiah	Eric	Jamey

Project Code:
https://github.com/PhoenixFire432/Photon2.git

Original Screencast:
https://youtu.be/nMBeejaD-xs

******New Screencast****:**
https://youtu.be/R4u2u2_EiM8

Photon Multiplayer Game Write-Up

Modifications of the Project:

Some of the graphical environment elements

~~Currently, we have a railgun, an animated velociraptor and a variety of blocks (wood/ice) used to build towers.~~ We now have two dinosaurs (Velociraptor and Pterodactyl) that can be choses to launch on the railgun. We also have completed blocks for building and a dinosaur egg that stacks on top. Along with a title screen and added music/sound effects.

Human Player

~~Currently, the player has been modified to spawn as a railgun which can shoot dinosaurs at the opponent's tower.~~ We now have both players connecting and able to choose either railgun/tower mode.

Win/Lose Condition

The player who selects to launch dinosaurs will win if they can knock down the egg within an x amount of dinosaurs being launched. Otherwise, they lose the game.

The player who selects to build the towers will win if their tower is still standing after the opponent has launched an x number of dinosaurs. Otherwise, they lose the game.

Member Contributions:

Jamey

- Model and Texture Variations of Boxes, Tall and Short (Stone, Wood, Ice)
- Model and Texture Egg
- Model and Texture Railgun
- Model / Texture / Rig / Animate Velociraptor

Bobby

- Setup Trello Board & GitHub
- Update GDD
- Manage meetings and document notes
- Found audio samples for potential use

Akiah

old

- Photon networking for room logic
- Projectile instantiation + physics implementation over network
- Rail Gun controls, steering, and Firing logic
- Player Role Logic (not complete)
- Camera movement logic
- Rudimentary UI and victory screen
- Game state logic

new

- Changes to cannon player
  - Cannon player can now select ammo type for each shot
  - Cannon player now has limited shots
  - Re-did ammo preview system
  - Re-did firing logic to work with new photon implementation
- Created Blocks player
  - Blocks player can spawn in new blocks up to a cap
  - Can move and rotate blocks with the space, left shift, and mouse wheel
  - Prevent cannon player from firing until blocks player gives the go-ahead
  - Implemented RPCs for the blocks to limit unexpected physics behaviours over the network

- General changes
  - Refactored player controller logic and game manager for easier maintenance
  - Implemented modular block and ammo templates, allowing fast additions of new types of blocks or ammo
  - Re-structured the scene hierarchy to support long(er) term development
  - Implemented win and lose conditions
  - More advanced UI with multiple panels that are enabled or disabled based on game state
  - Overhaul of entire networking implementation
    - Properly implemented photon for cannon and blocks players
    - Got RPCs working for their intended purpose (win/lose, sending data, ensuring only one player can be a blocks player or a cannon player)

Eric
Old

- Tower Instantiation + physics and colliders
- Significant structural overhaul for tower
- Singleton and RPC research
- Audio Manager and audio clips playing
- Audio over network (not complete)
- Programmatically organized Velociraptor animator

Member Reflections:

a. **What considerations you must make that are different from single player games when you design and implement a multiplayer game.**

b. **How multiplayer gameplay changes the nature of a game from a theoretical perspective.**

c. **How a multiplayer environment might be created where learning could occur, the nature of that learning**

d. **how you would go about implementing such an environment.**

Jamey

A. **Multiplayer games are more conscientious where things are located in folder systems as well as programmatically. As someone who worked more on the assets, I didn't get as much of the hands-on information, but from what I**

learned of python and input from other groups I observed that assets across the server are very particular so that they may be consistent across each person's device.

B. Multiplayer games are undertaken in a group and become a social activity, whether it be through in game chats, pvp modes, or collective efforts. This changes the dynamic of a gameplay and even subsequent behaviors of players. The environment can become more lighthearted or competitive, versus when you play solo it is usually the same dynamic of Player vs. Game. In multiplayer games, this can range significantly.
C. Learning as a group skill is something GIMM aims to foster, because in a group applications of knowledge even to gameplay scenarios increase opportunities for learning due to diversity of experience and opinion. There are environments beyond gameplay that are being created, such as shared creative spaces over Hololens (Microsoft Mesh). Whatever the situation is, be it lighthearted gameplay or collaborative sessions, being with other people is bound to expand the experience and open the opportunities for learning.
D. Implementing would be different depending on the scenario. For something more lighthearted and fun, I think it would be useful to establish a team-building structure, one where roles and multiple members are tangibly important to the learning and progression of a world. Logic and skill based operations can be implemented in game-like formats, and coming up with multiplayer puzzles could accomplish this.

Bobby

A. You need to have a server which can communicate across multiple players and platforms. It also must be able to update the different changes that each player has on the environment and reflect those changes in real time.
B. Multiplayer games allow you to interact with other players. This is different from single player where the player usually plays with/against an AI. I know from the Hummingbird Project that AI agents are capable of learning quickly and can become pretty godlike in video games. So, playing against other players (multiplayer) might be a better option for someone who just wants to play for fun vs. Playing against an AI agent (single player).
C. Multiplayer games have been packed with learning in my experience. They create an environment where you could work as a team and teach one another different ways to tackle an objective of the game. I think the nature of learning comes from practice and experience. Tons of people watch YouTubers play games so they can learn from them and the way they play. The same can happen live in multiplayer video games.
D. I would set up an environment that would give the players a common goal. Then I would provide tools/actions that could be used to reach that goal and allow others to learn from each other on how they use those tools/actions.

Dead By Daylight comes to mind when I read this question. They set up an environment where a team of survivors must use their tools/skills to repair 5 generators which will allow them to open the exit gates and leave. Someone could choose to focus more on repair work, others could focus more on distracting the killer. Either way there are always new ways of playing to learn and adapt to.

Akiah

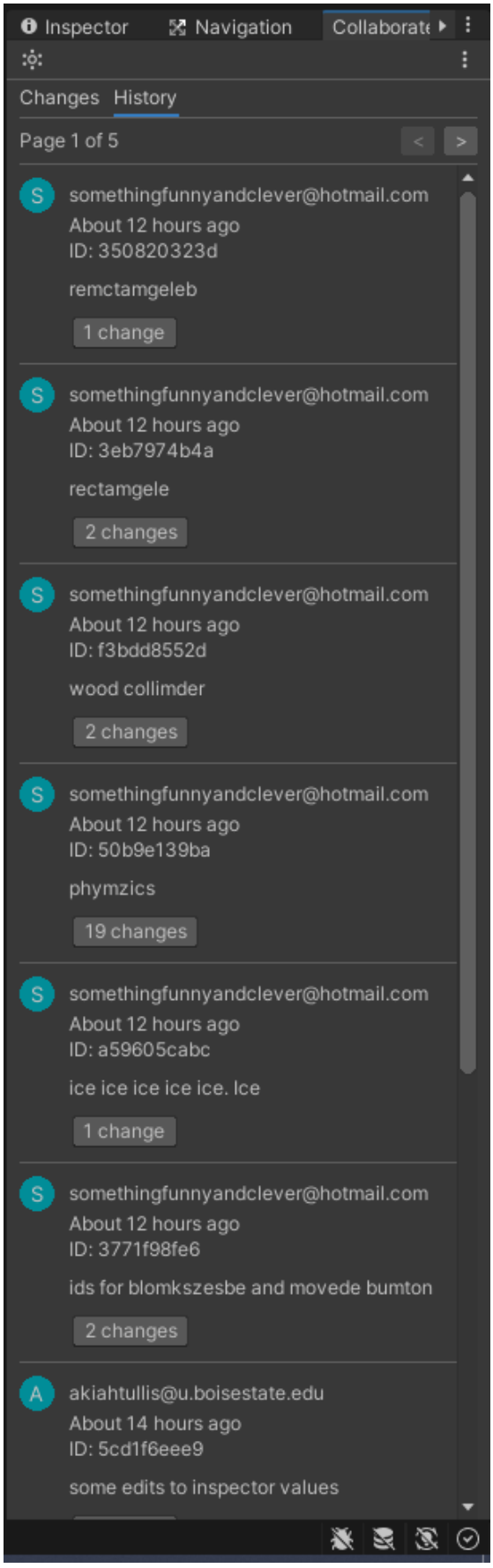
Notes from January 2022: I still stand by all of this. One thing to note is that I did get a chance ot overhaul the project, and I think I learned a lot in the doing of it.

A. **There are huge structural considerations. The big difference is that you need to divide the game into networked and non-networked sections, while allowing them to interact. If we were to do this project again there would be major changes in structure. Even without re-doing the project, I found myself frequently re-factoring as we learned more about networking and Photon.**
B. **Games are experiences. A multiplayer game is a shared experience. Although there are parallels between the experiences of two people who play the same singleplayer game (just as there are for two people who read the same book), they just don't have the minutia of details that come from interaction with another. Whether the nature of a multiplayer game is competitive or collaborative, it is ultimately about interactions with other players. A singleplayer game, on the other hand, is ultimately about interacting with the vision of the creator(s).**
C. **Learning is (generally) mimicking the behaviors of those around you, who you expect to behave in more adaptive ways (although learning can happen without experts to observe, it is slower). Multiplayer environments can provide a 'testing ground' to observe 'correct' behaviors or to experiment/practice with them. Learning almost always occurs in multiplayer games, as they are a collaborative effort to improve. Competitive games have wikis and guides, collaborative games have mentors and communities. Whether this learning is useful outside the scope of the game, however, varies.**
D. **The greatest considerations, in my view, are 1: mentorship, 2; fidelity/sensory feedback, 3: interactivity, and 4: low-stakes. Mentorship can be formal or informal (although formal is usually better to ensure that it gets done). Proper mentorship can ensure that learners know what they ought to try to emulate. High fidelity/sensory feedback allows learners to properly understand the way that their mentor(s) interact with the environment. This allows for emulation. High interactivity allows learners to experiment with behaviors without leaving the platform. This is important in order to 'tighten the loop' that refines learned behaviors. Finally, lowering the stakes encourages play, which in turn is a state that encourages learning.**

Eric

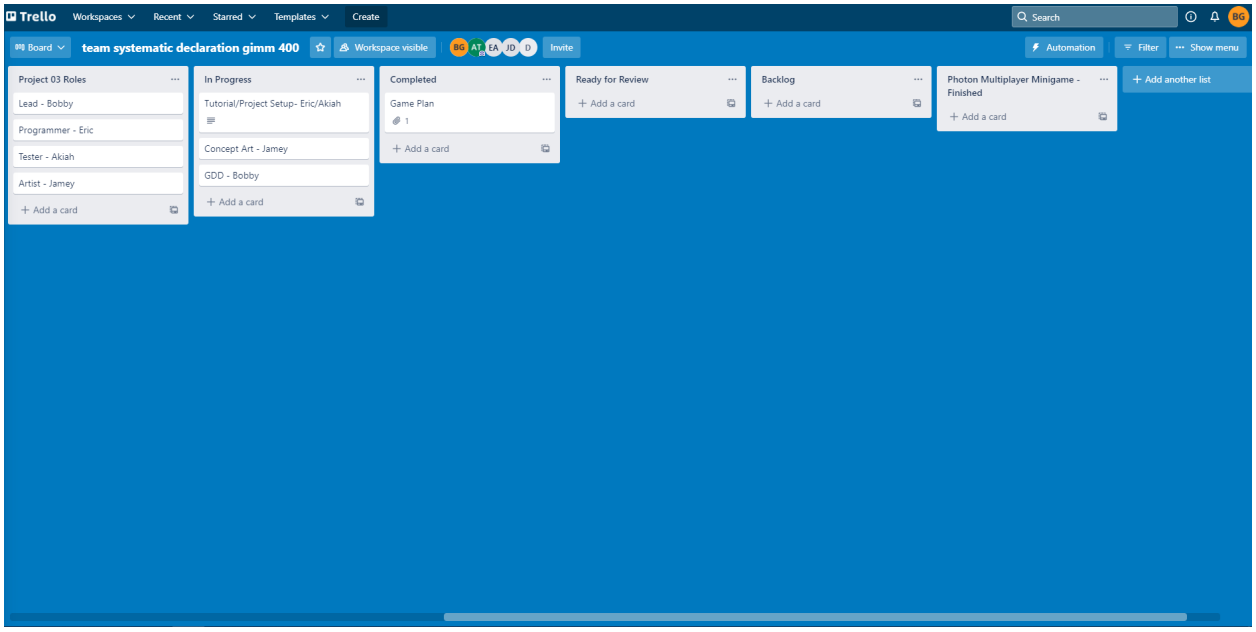
A. **You definitely have to take into account that networked code is far different from local code. You have to keep in mind that some things are local and some things are networked and consider what goes where, as well as how to implement it and how you want it implemented over the network.**
B. **Multiplayer adds an entirely new thing to account for another player. You can't just optimize everything with the expectation of their being only one player to optimize for. You now have an entire other person who will be playing at the same time as another player and you have to figure out how to give them similar experiences and not break either one for the other.**
C. **Learning is done best through trial and error, so implementing a multiplayer learning experience I would imagine would be set up to reflect that. With either you and your friends given tools to mess around with or given detailed instructions doesn't mean much. The actual learning bit comes from trying, failing, trying again and learning why things are happening the way that they are and how.**
D. **I would implement such an environment by attempting to give the players tools to mess with and an open space to explore. Older Minecraft is a solid example of this, you had a crafting table, but you didn't know any recipes to craft anything natively. You had to learn from friends in the game with you or learn from it's online community. Eventually you get a large amount of knowledge gained from the efforts of many working together to figure things out. "How does redstone work?", "How do I craft a(n) [x] item?". Being there with a friend allows you to mess around and learn, or you could watch tutorials posted by the online community to learn from them.**

Version Control Verification:

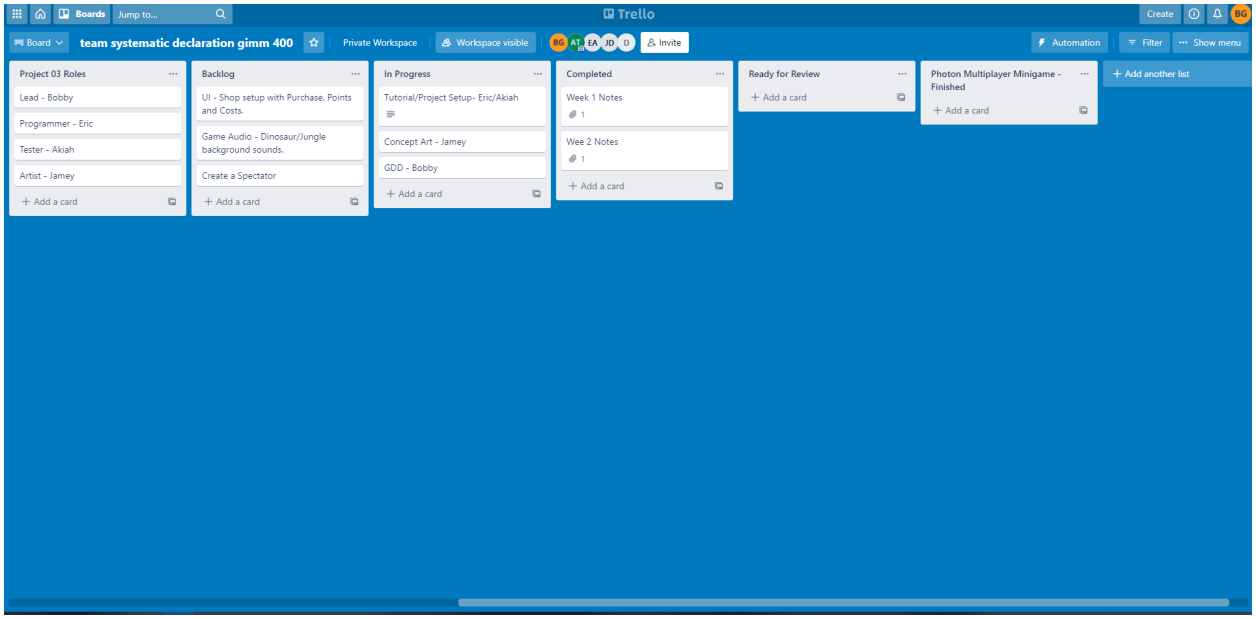

Inspector
Navigation
Collaborate
Changes
History
Page 1 of 5
<
>
S
somethingfunnyandclever@hotmail.com
About 12 hours ago
ID: 350820323d
remctamgeleb
1 change
S
somethingfunnyandclever@hotmail.com
About 12 hours ago
ID: 3eb7974b4a
rectamgele
2 changes
S
somethingfunnyandclever@hotmail.com
About 12 hours ago
ID: f3bdd8552d
wood collimder
2 changes
S
somethingfunnyandclever@hotmail.com
About 12 hours ago
ID: 50b9e139ba
phymzics
19 changes
S
somethingfunnyandclever@hotmail.com
About 12 hours ago
ID: a59605cabc
ice ice ice ice ice. Ice
1 change
S
somethingfunnyandclever@hotmail.com
About 12 hours ago
ID: 3771f98fe6
ids for blomkszesbe and movede bumton
2 changes
A
akiahtullis@u.boisestate.edu
About 14 hours ago
ID: 5cd1f6eee9
some edits to inspector values

Trello Updates

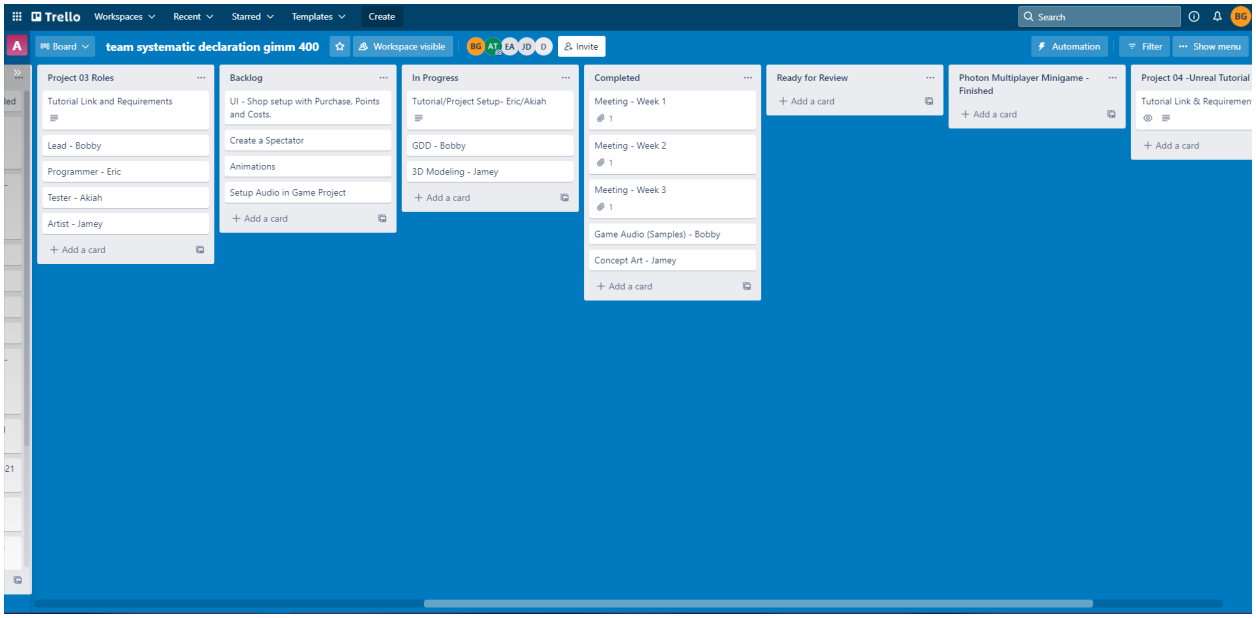
Week 1



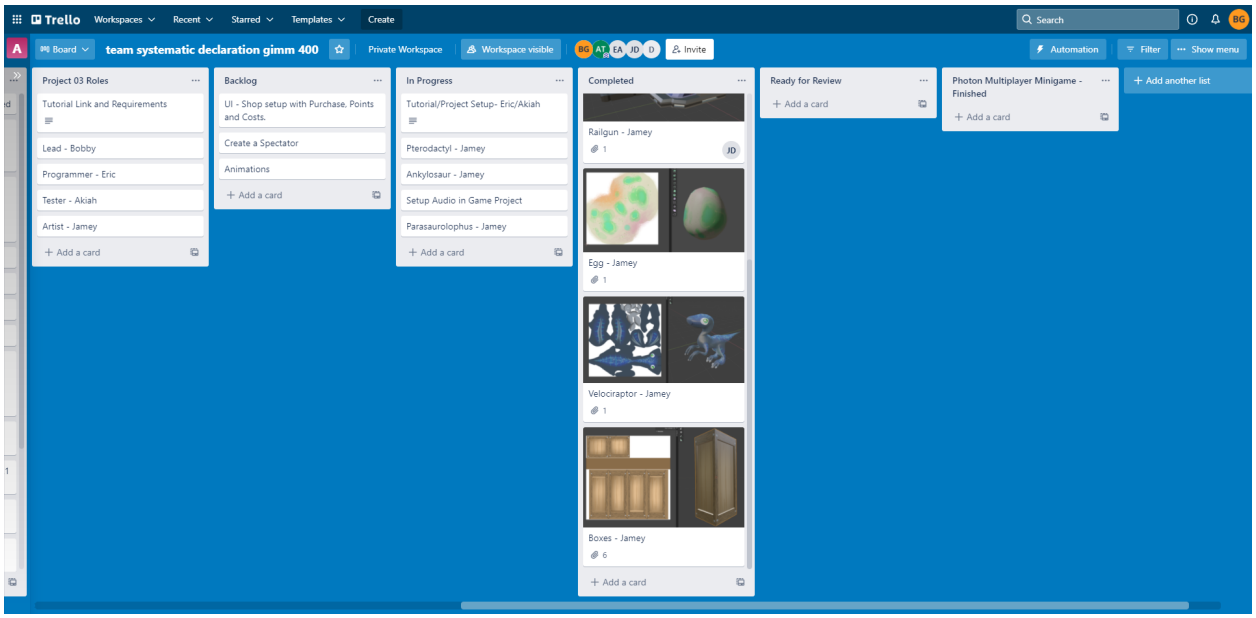
Week 2



Week 3


Trello
Workspaces
Recent
Starred
Templates
Create
Search
Board
team systematic declaration gimm 400
Workspace visible
Invite
Automation
Filter
Show menu
Project 03 Roles
Tutorial Link and Requirements
Lead - Bobby
Programmer - Eric
Tester - Akiah
Artist - Jamey
Add a card
Backlog
UI - Shop setup with Purchase, Points and Costs.
Create a Spectator
Animations
Setup Audio in Game Project
Add a card
In Progress
Tutorial/Project Setup- Eric/Akiah
GDD - Bobby
3D Modeling - Jamey
Add a card
Completed
Meeting - Week 1
Meeting - Week 2
Meeting - Week 3
Game Audio (Samples) - Bobby
Concept Art - Jamey
Add a card
Ready for Review
Add a card
Photon Multiplayer Minigame - Finished
Add a card
Project 04 -Unreal Tutorial
Tutorial Link & Requiremen
Add a card

Week 4


Trello
Workspaces
Recent
Starred
Templates
Create
Search
Board
team systematic declaration gimm 400
Private Workspace
Workspace visible
Invite
Automation
Filter
Show menu
Project 03 Roles
Tutorial Link and Requirements
Lead - Bobby
Programmer - Eric
Tester - Akiah
Artist - Jamey
Add a card
Backlog
UI - Shop setup with Purchase, Points and Costs.
Create a Spectator
Animations
Add a card
In Progress
Tutorial/Project Setup- Eric/Akiah
Pterodactyl - Jamey
Ankylosaur - Jamey
Setup Audio in Game Project
Parasaurolophus - Jamey
Add a card
Completed
Railgun - Jamey
Egg - Jamey
Velociraptor - Jamey
Boxes - Jamey
Add a card
Ready for Review
Add a card
Photon Multiplayer Minigame - Finished
Add a card
Add another list

***Photon Extension Member Contributions:

Jamey

- Model / Texture / Rig / Animate Pterodactyl
- Implemented new skybox

Bobby

- Created artwork for Title Screen
- Created audio track for title screen + button audio
- Update GDD

Akiah

- Changes to cannon player
  - Cannon player can now select ammo type for each shot
  - Cannon player now has limited shots
  - Re-did ammo preview system
  - Re-did firing logic to work with new photon implementation
- Created Blocks player
  - Blocks player can spawn in new blocks up to a cap
  - Can move and rotate blocks with the space, left shift, and mouse wheel
  - Prevent cannon player from firing until blocks player gives the go-ahead
  - Implemented RPCs for the blocks to limit unexpected physics behaviours over the network
- General changes
  - Refactored player controller logic and game manager for easier maintenance
  - Implemented modular block and ammo templates, allowing fast additions of new types of blocks or ammo
  - Re-structured the scene hierarchy to support long(er) term development
  - Implemented win and lose conditions
  - More advanced UI with multiple panels that are enabled or disabled based on game state
  - Overhaul of entire networking implementation
    - Properly implemented photon for cannon and blocks players
    - Got RPCs working for their intended purpose (win/lose, sending data, ensuring only one player can be a blocks player or a cannon player)

Eric

- Reimplemented the audio to work with the new changes
  - New scripts means relocated lines!
- Added new sounds for the second dino
  - Pterodactyl has their unique sounds
  - Both dinos sounds get quieter over the duration of them being launched

- Added sounds for collision
  - All of the blocks make sound when colliding with each other (plays two different sounds if they are two different block types)
  - Blocks make their respectives sounds when colliding with the floor and dinos
  - Dinos make their colliding sounds when they hit the floor, each other and blocks to play their respectives sounds
- Got the second dino being shot out correctly with correct meshes
- Filmed and edited the video
- Got a coroutine set up to add a one second delay to one of our checks
- Got all of the sounds to play correctly over the network
- Led the testing
  - We had a lot of interesting issues with testing, like how the check to see if the builder player won. It would check to see when the velocity was 0. Which it did successfully, however, the frame the ammo was spawned to be launched "technically" has a velocity of 0. Meaning the builder would win the second the last shot was fired, instead of after their shot cannot move. This led us to implementing a coroutine to implement a one second delay for that check. The builder player would now win correctly, after the last shot fired had finished its movement.
  - Also led the testing to make sure all of the sounds were correctly playing over the network for both players