Production II Project

Technical Risk Assessment

(Jak Tiano, Ian Sartwell, Ryan Leslie, Shain Strother, Evan Schipellite, Tim House)

Prepared by: Evan Schipellite

Introduction

The purpose of this document is primarily to provide a detailed outline of the project requirements while simultaneously providing analysis based on the potential risks relating to those particular areas. In this sense, this document should not server to portray to complete feature requirement list, but rather introduce generalized areas of interest to better understand the programming perspective, scope, and uncertainties within the project. This should allow members to review the project's requirements and direction, as well as reach a consensus on the importance of features in relation to the potential risk and time expectations surrounding the development of those features.

The project itself is, as discussed further in the Technical Specification and Design Documents, based around the concept of allowing two to four players experience gameplay on a single screen. This immediately removes the concern for online networking, but still enables the reality that the development, balancing, and testing of local multiplayer features may require some additional research. This may not be overt at the start of the project development, but after Alpha it may become more an issue when testing is primarily based around user feedback. The game will create a simple menu environment that allows players to adjust simple features relating to sound and controls. Upon beginning gameplay, the players will encounter an inventory selection screen to create their robot and begin gameplay.

Although an initial phase involving the collection of inventory parts may have been originally planned, depending on development progress, the feature may be discarded in favor of polishing mechanics. Regardless, the requirements and assessment for the first phase will be discussed further on in this document. After selecting their inventory, players will continue gameplay by entering a combat zone. The last player remaining wins the round, and players will be returned to the start menu or inventory selection screen depending on user feedback. Therefore, the initial understanding of the game can be best understood as relying largely on UI elements, player feedback, local multiplayer compatibility, and mechanic gameplay functionality.

Therefore, this document will server to list topics of interest, discuss their overall risk expectations, and then further elaborate on details regarding subtopics if applicable.

Local Multiplayer

The game will require two to four players for its experience to be conveyed. Due to the limitations of the keyboard, controllers will be utilized for playability. While the current target platform is specifically designed for PC release, eventual development is understood to be directed toward the Xbox Live Arcade and the Ouya. In both these instances, the standardized controller design will assist in ensuring that all players compete with similar controller designs. With that understanding, it is best to further evaluate the specific areas relating to the multiplayer structure.

Standardized Input (High Risk, High Time Allowance)

While mapping a single controller to gameplay in Unity does not prove difficult, in order to test with multiple controller types and standardized system or external library will need to be created or located to improve production quality. Unity recognizes Xbox controllers and PS3 controllers as separate identities, and therefore axis mapping is not shared between the two controller types. Xbox controllers treat the back triggers as a single axis, while further prevents the controllers types from being treated similarly. In other words, even if a simple DeviceProfile class was created, specific tests would still need to be conducted to convert an Xbox trigger input from an axis to a button press.

The most promising solution to these issues relates to the research and implementation of an external standardized library already available for Unity. Patrick Hogan’s InControl Unity Manager may prove useful in terms of mapping standardized input. However, the process of implementing its features, debugging any problems that arise, and maintaining the external challenges arise as a problem in terms of keeping it functioning as the project continues onward. Due to the nature of various controller types, unless a specific controller type is dictated by the Design team, the research and implementation of a standardized library will definitely be time-consuming to apply and may cause further issues later on.

Control Mapping (Low Risk, High Time Allowance)

While not technically difficult, the control scheme will need to be easily adjustable. It is likely that the manner of controlling the player’s mechanics will adapt and evolve over the course of the project. Therefore, instead of redesigning the entire input process, the button declarations should be easily swappable. This may come with the development of the standardized input system, but it worth noting since a decent amount of time may be applied in order to test control schemes and evaluate vector math for various axes.

Controller Connection (Medium Risk, Medium Time Allowance)

The game should be able to adapt to any circumstances regarding device counts detected. Therefore, the game should only allow for the available number of controllers connected.

Upon disconnect, the game should notify the player about the disconnection and proceed to await a reconnect. In this sense, the game should consistently be evaluating the device count and be ready to assign or remove devices depending on the controller states. This may not prove difficult, but may take a bit of time to test and cover the anomalies.

Menu System

The game should convey a relatively simple menu system for players to navigate through. This system will primarily be navigated by the first player (the host) and it should allow them to play the game, search through options, and exit gameplay. The art direction will be primarily determined over the duration of the project, but the functionality should remain the same over the course of the project. In comparison to other topics of interest, this area represent a section that should primarily have low risk with medium time allowances, mainly because menu creation should be a simple procedure. The only real risks relate to the process of learning of to transfer menu creation knowledge to the Unity scripting database.

<u>General Menu Layout (Low Risk, Low Time Allowance</u>**)**

The overall menu should just be a matter of navigating through GUI elements or planes within Unity. The only real time-consuming components relates to the development of the Options menu. This menu will need to boast some basic features such as sound levels and possible adjustments to improve performances for various platforms. The expectations may be set later, but it is likely that the general menu system will be easy to implement at the start of the project, but will likely require further addition of features later. Therefore, there's a need for the menu system to remain flexible throughout the project duration.

<u>Inventory Selection Screen (Medium Risk, Medium Time Allowance</u>**)**

The inventory selection screen should enable the users to select their available robot parts and 'ready' their slice of the screen to prepare for the actual gameplay. Creating the functionality for part selection and readying should be simple enough. The complexity, however, can be found in two specific areas. First, the inventory selection screen should provide a model viewer. Obviously, the layout of the menu should be created with collaboration from the Art team, but the procedure of bringing the player model into the selection process rests primarily with the Programming team. This should just be a matter of creating a temporary player object and settings its parts, color, and other features. Yet, there are other unknown problems that could result from this feature. Second, the other complexity relates to the testing for controller connections / disconnections. This is mainly a matter of testing, but some errors may occur in bizarre situations, such as when a player is ready but disconnects their controller.

Game Over Screen (Medium Risk, Medium Time Allowance)

The end screen for matches will likely need to display scores and game info to the players. This info should easily be saved over the course of the combat experience, and therefore will acknowledge damage totals and health statistics. However, from this screen the players should be able to select a restart, return to the main menu, or return to the inventory selection screen. This will also require some testing and design feedback, as well as some UI design to properly create the functionality for the menu components. In other words, while all of these features are simple to add, allowing them to exist without hindering the visuals of the end screen will take some collaboration with Artists to create the structure.

Player Architecture and Mechanic Creation

The architecture style, relating to the code units, for the player components certainly remains as the foundation for this project's gameplay. Every player, while maintaining the same robot base class, will then essentially utilize a polymorphic system to add scripts that represent different types of parts for the robot's torso, arms, and legs. Due to previous programming knowledge, the actual setup for this scripting architecture should not be an issue. The most prominent risk relates to the actual creation of abilities for the types of parts, as well maintaining the flexibility of the code units.

Player Base (Low Risk, Medium Time Allowance)

The player or robot base will essentially need to be able to load up the torso, arms, and legs scripts, along with their corresponding model pieces. As long as the previous menu is able to transfer data representing the chosen parts of each player before the combat experience of the game, the robot base class should be able to easily assign the appropriate parts using the Reflection method to acquire classes, models, and other aspects that all share the same declaration or names. It may take a bit of tweaking to increase flexibility, but the player base shouldn't take too much time to originally create and it should also be relatively easily extensible.

Mechanic Creation (Medium Risk, High Time Allowance)

As noted beforehand, since all parts will contain a corresponding action, the polymorphic structure should allow for the creation and maintenance of numerous robot part types. A simple system to place the models for the mechanic parts will need to be created to connect joints to the torso piece, but otherwise the code will largely depend on the type of mechanic being implemented. Therefore, while implementing new mechanics should prove easy, the actual risk and time allowance depends entirely on the actual mechanics being added to the gameplay. Therefore, in order to further elaborate and assess the mechanics, a list of current mechanics will follow to briefly describe the nature of the mechanic in terms of risks and time constraints.

- Legs
    - <u>Tank Treads (Low Risk, Low Time Allowance)</u>
        - This feature will allow for players to passive damage enemies, as well as deal extra damage on active. Unless this feature undergoes further redesign, it's foundation should be relatively simple.
    - <u>Jetpack (Medium Risk, Medium Time Allowance)</u>
        - The Jetpack legs might provide an issue in terms of creating it in a manner that is easily understood by the players within the game. The leg feature should promote the idea of allowing a player to hover, move, and damage players below them.
        - However, a cooldown feature to limit excessive use will need to be tested and implemented. A simple cooldown system would present an issue where a player only hovers for a moment before being interrupted.
        - A unique cooldown system that is only set back by the duration in which the player has used it may serve as an alternative that can be used through the model.
    - <u>Spider (Medium Risk, Medium Time Allowance)</u>
        - Allowing the player to dive toward the ground and cause an explosion should be relatively simple. However, consistently adding a downward force and creating the explosion particles may take some time to perfect.
- Torso
    - <u>Body Slam (Medium Risk, Medium Time Allowance)</u>
        - The body slam could cause some bugs with regards to the collision detection for damage and forces. Due to the nature of physics, the method for determining the duration of the potential damage time after utilization may vary depending on the player's speed. Therefore, it should be possible to calculate their velocity, and once it drops below a threshold, turn off the body slam feature. This could cause other issues, so this mechanic should be monitored and tested thoroughly.
    - <u>Laser Beam (Medium Risk, Medium Time Allowance)</u>
        - The laser beam ability should be designed to launch a buffet of energy at players within range. Ideally, it should arc and cover the entire area in front of a player, thus allowing them to knock-back and damage enemies.
        - This mechanic may cause some issues in terms of balancing damage to ensure its usefulness when the enemy is close and at a distance. In other words, the enemies should experience meaningful punishment for getting hit by the beam at any distance, even if they take increased damage when within close range.

    - The other concern relates to how the laser beam will be rendered, as a trail renderer should be utilized to create the actual beam effect. Due to the fact that the Programming team isn't too experienced with these shader types, this may take a bit of time to research.
  - <u>Energy Shield (Medium Risk, Medium Time Allowance)</u>
    - The energy shield should knock-back and damage nearby enemies. This feature alone isn't too difficult to do, but adding the particle effects and reflecting enemy attacks may take a bit of time. If anything, reflection may be limited to projectiles in order to balance and scope the creation of the energy shield.
- Arms
  - <u>Sword (Low Risk, Medium Time Allowance)</u>
    - The sword should just be a matter of moving the player's arms and creating an invisible damage field within that area. Therefore, testing should bring about any bugs with the sword, but the implementation remains relatively easy.
  - <u>Fists of Fury (Low Risk, Medium Time Allowance)</u>
    - The main concern for the fists is to ensure that the gameplay is distinguished from the sword. While the sword allows the player to strike for massive damage, the fists should represent a faster weapon with less damage.
    - In this sense, it is much like the machine gun in the manner that it provides the player with more time to move around in favor of quick damage and additional effects.
    - With each successive hit within a time limit, the player can deal extra damage, thus encouraging the player using the fists to stay within range and jump around their target to continuously harm them.
  - <u>Machine Gun (Low Risk, Medium Time Allowance)</u>
    - The machine gun is mainly a matter of creating a burst-fire weapon that shoots a number of projectiles that hit for relatively low damage. The main challenge is a combination of balancing and tweaking the trail renderer for the visual representation.
  - <u>Rocket Launcher (Low Risk, Medium Time Allowance)</u>
    - The rocket launcher essentially provides the opposite feature when compared to the machine gun. It fires a single projectile at a delay that explodes upon impact. Once again, creating the trail renderer represents a primary issue. Also, the rocket launcher should home in on targets in order to increase its effectiveness during the fast-paced gameplay. This homing feature should be available for all projectiles, and should be variable in order to ensure that it only slightly adjusts the rocket's course.
  - <u>Riot Shield (Low Risk, Low Time Allowance)</u>

        - The riot shield mainly represents a mechanic that should be able to damage enemies like the sword, but also able to reflect and convert projectiles on action hit. While these features may sound complex, they will mainly derive from the reflective nature of the energy shield and the attack pattern of the swords, and therefore shouldn't take too much time to implement, test, and perfect.
    - <u>Grapple Whip (High Risk, High Time Allowance)</u>
        - Although likely the most unique mechanic, the grapple whip does represent the trickiest feature to implement. The player should be able to fire a grapple that should hook an enemy player and bring them to the owner. This involves potentially hinging the grapple system to attach itself to another player and then translate the player back to the owner. This may not prove difficult in the long run as a prototype effect, but the process of perfecting and making it appear believable will likely take a lot of testing and redesign.
- Projectiles
    - Projectile-based weapons will have their projectiles derived from a base class. This class should contain simple information regarding the projectile damage, its own health, its owner, and other general features shared by all projectile types.

Physics System

The physics system will need to constrict movement to a 2D space, while also allowing planets to exhibit orbital systems that attract players, allow them to orbit around their structures, and enable players to leap between planets and undergo the combined forces. The process of creating, maintaining, and tweaking this system will likely represent the most difficult part of this project, as the development of the physics system will ultimately impact the method in which the various mechanic parts will interact with the players and the world. Therefore, before the majority of the mechanics are implemented, the game should be able to effectively allow the players to simply jump and navigate around any number of planets without any game-breaking bugs or overt necessities to alter the features of the physics system.

<u>Gravity Sources (High Risk, High Time Allowance</u>**)**

Corresponding planet structures should have gravity source scripts attached that allow them to have a collider detectable by the players. Using this information, players can inherit gravity object scripts that allow them to undergo forces toward the center of the gravity sources when within range. The development of this feature along with the many bugs that will be discovered during gameplay testing places it at high risk, as it will not only take a large amount of time to perfect, but the methods for applying gravity from various sources will require a number of different architecture designs to fulfil..

Ideally, it may be best to have players detect a gravity source within range and undergo the center force from the nearest gravity source. If a player is encountering two gravity sources within range, they will opt to fall toward the initial one detected. Once they leave that object's range, they can then undergo the force from the other gravity source. If an event occurs when a player leaves the range of a gravity source, but there are no other sources nearby, they will simply retain their connection to the last known source and fall back to that source. This should allow for players to stick to planets. A toggle feature while in the air can be utilized to enable players to switch gravity orbits, thus allowing for more control of movement, jumping, and orbiting.

UI Elements

Some UI elements, relating to player health and cooldown conditions, should be noted here to discuss the complexity and time behind the design of these features. While technically not difficult to implement, the process of adding them in a local multiplayer environment limits the visibility or secrecy of some of these elements, as all players will be struggling to view the same screen and monitor their character at the same time. Therefore, it is best to extract design aspects in advance in order to test a number of different UI designs during the project in order to find the best manner for represents cooldown aspects for the player parts, the player health, and other possible gameplay elements that might convey events in the level or potential time limits.

<u>Camera (Low Risk, Medium Time Allowance</u>**)**

The camera should be able to keep all current players on the screen while maintaining a minimum and maximum zoom. The camera should further clamp accordingly in order to ensure that players leaving the gameplay area do not further cause the camera to zoom out. An art element to show the direction and location of off-screen players can also be implemented to allow for visual assistance. The camera itself shouldn't be too difficult, as it primarily relies on position averaging and vector calculations to determine how the camera will proceed to look at the scene. But, some bugs may occur that will require further iteration on the camera design.

<u>Health and Cooldowns (Low Risk, High Time Allowance</u>**)**

The players will likely be concentrating on their characters for the duration of the game. Still, some visual representation for health and cooldown times should be displayed to the players. Whether this is on the sides of the screens, the top of the player, or built into the view of the model parts, the reality is that the aspects will not be particularly complex. Still, collaboration with the Design team will allow for testing sessions to locate the best possible methods for conveying player health. It is likely that the health bar can be provided above the player. By following a point that follows the player, it can be positioned in a manner so that it never rotates, thus preventing the player orientation from influencing the view of the health bar. Otherwise, cooldown icons will primarily only relate to the torso and leg abilities, as the arm abilities are much or ‘spammable’. In this sense, it may be best to create a shader or particle effect based on the torso and legs to display readily to allow players when the mechanics are functioning.

Stretch Goals

While the previous features relate to the currently defined and expected mechanics, some other features have been discussed for the project that can be best defined as ‘stretch’ features. In other words, these features reflect aspects of the game that, while possible additions, are not required for the final and complete game experience. Nonetheless, it is worth mentioning and analyzing a few of them in case the project diverges in a direction that may eventually require them over other features, or if time is available for the implementation of some of these features. If anything, the risk analysis for these features should server as a caution and detailed supply of evidence concerning why these features have been pushed to the side and what slots of time and research would be required for their implementation. Rather than simply going ahead and integrating these features at the end of the project, the analysis should assist in providing caution for their explorations, as a number of uncertainties may follow alongside the implementation of some of these extra aspects.

First Phase Gameplay (Medium Risk, High Time Allowance)

Originally a major addition to the gameplay, the first phase reflected a brief mode where players would compete against each other to collect robot parts. Their selection screen would then be limited by the parts collected. While simple in theory, progress on the main combat gameplay and the selection screen has allowed for the concentration of the game to switch to the overall choices made by the players for each combat round. Since the game can fully function without the first phase, it has been considered a good idea to simply focus on the implementing of core features and leave the first phase as a possibility if time allows.

Overall, the first phase may not be overly difficult to create the functionality for. Players would simply be able to compete against each other by navigating on 2D planes representing the cut components of planets. Part icons would randomly spawn within the planet areas during the time duration. Players would then be conveyed the parts collected on the inventory screen. Players missing parts would be provided with a default set. Essentially, the major risks for the gameplay fall in three areas.

First, development of the default mechanics would be required for this phase to properly work. These mechanics would be simple attacks with the default pieces, and therefore would not be too time-consuming to implement. However, depending on the time left within the project, even these might be too much. Second, the level design would have to be created in a manner that allows all players an equal chance to find and collect items. This means that about one to two weeks would need to be available to test the first phase gameplay. Third, the art direction has desired additional visuals for movement in the planets requiring voxel components. Unless this aspect is dropped in favor of simple 3D drill models on 2D cut-out planets, the creation of this feature would take at least three weeks to finalize.

Player Eject on Death (Medium Risk, Medium Time Allowance**)**

The aspect of player death initially reserved the idea of having the player spawn from their robot open death. Currently, the game works well with players being destroyed on health loss. However, due to the party-game direction, player personnel exiting the robots would provide an extra step to finish off players and create immersive gameplay.

Despite this reality, it is highly unlikely that this feature should be added within gameplay unless an excessive amount of time remains at the end of the project cycle. While the functionality of the feature would not be too difficult (spawning a new player object on death), it would require art, animation, and extra design decisions that would impact gameplay. In many ways, this risk isn't particularly related to how the programmers would manage the features, but from a perspective of scope, the amount of testing to create animated characters and design gameplay to interact with the second step in player destruction would be too time-consuming and ultimately take away from the more important testing of robot parts and levels.

Player Profiles (Medium Risk, Medium Time Allowance**)**

An interesting option would be to allow players to select or add their own profiles during gameplay. In doing so, the game would remember their player names, color choices, and statistics for each round. Information would be saved on the local device, thus allowing players to continue their character progress. Since this is mainly a vertical slice, this feature is not required in the current prototype stand-point, but some of its aspects could be examined for similar development. In other words, the entirety of this feature will probably not be added, but aspects such as player color or name tags could be added to improve the immersion in gameplay. On the inventory selection screen, players may also have the option of adjusting their color or choosing from defined player names that can be added or removed in the options menu.