Mission Impawsible: Technical Document / Risk Assessment

EGD220: Project Team 4

(Caitlin Coon, Nicholas Robison, Noah JonesDrayton, Evan Schipellite)

Prepared by: Evan Schipellite

Technical Synopsis

The goal of this document is to introduce the programming language, personnel, estimated work amounts, and overarching mechanics of the game in creation. Sections will be divided according to the programming requirements and the expected content to be included. Further subdivisions will dictate the actual risk assessments for various milestones, while also potentially submitting a potential list of features to be added during that milestone. Content included in this technical document are subject to change as a result of team meetings or complications during the process, however the ongoing list of potential risks will assist in limiting the amount of complications that may occur by addressing them in an earlier time frame.

As it stands, the current basis for the game in creation is a one button application based in Actionscript 3. Further explanation will be provided in the concept and design statements, and therefore this document will only focus on the actual mechanics being analyzed by the team. The game will involve the movement of a player character on a pre-set line, allowing the player to overcome various obstacles through proper time and / or tapping of the button as they attempt to make their way through the path to a determined objective. As a result, much of the technical document will be concerned with the creation of the various aspects of the line, movement, and mechanics. Much of the foundation mechanics will be created in the second milestone, while the polishing additions will be added and tested during the third milestone. Although much of the game mechanics are not expected until the second milestone, many of the core aspects of the game will be developed during the first milestone to allow for flexibility in level creation.

Programming Requirements / Further Introduction

The entirety of the project will be designed and written in Actionscript 3, utilizing FlashDevelop for code structure. The project will not only seek to include the design mechanics in an efficient and flexible manner, but the code should also be somewhat understandable by the designer such that the process of level creation or object placing can be interpreted by non-programmers through the use of text files to input coordinates. At the very least, the program should have a feature to allow the designer to build and test levels through text files and running the application. It is expected that all of the art and designs assets will be implemented by the programmer, but communication will be required in order to format sprites and images collected.

Code should exhibit a defined class structure, and various mechanics and features should be developed in an efficient and universal manner to allow for swift changes in sizes, speeds, visuals, etc. While the programmer is expected to be the team member spending most of the time with the actual project, various aspects should still be commented and variables should be named properly in order to allow for others to comprehend the code if required. Various aspects to be developed will include the path / line feature, the player movement, menus, collision, various mechanics, and art implementation.

It should also be noted that risk be will divided in terms of **Low**, **Medium**, and **High** classifications. **Low** representations a feature that shouldn't cause any unexpected issues, but still is worth mentioning as far as features may be presented. **Medium** represents an implementation that may involve some research or testing, but its overall implementations should be manageable if given the proper time frame. **High** represents an aspect of the project that should be researched and tested as soon as possible in order to further evaluate its difficulty. It may require removal or altering in order to allow it to fit in the proper time frame, or it may simply require a large amount of work.

The intention of the rest of this document is as follows. This document will divide the following sections per milestone examined. Each milestone will be separated into two sections: risk expectations and risk reviews. At the start of each milestone a section will be created in order to list the various components expected to be completed, noting the estimated time and research required for each class / implementation. Afterwards, a short section will be created to list out the expected areas for risk in terms of time or difficulty. At the end duration for each milestone a short section will be created to convey the current standing of the technical aspects of the project, thus showing any complications that may have occurred or additions that were created for further group review.

Milestone 1: Risk Expectations

For this milestone, the programmer is mainly expected to meet with the group and develop the technical risk document. However, it may also be possible to begin developing a framework and prototype while also working with Flash. However, the only pending necessity is the technical risk assessment along with group cooperation.

Required Components

- The Technical Risk Assessment Document
  - Should simply be a structured risk / report to convey the programming aspects of the project for each milestone.
- Group meetings in order to acquire the expected mechanics and layout for the eventual game.

Optional Components

- A prototype for the game
  - Would include
    - the line path creation system
    - A moving player icon
  - Should essentially convey how the player would move along the path with the pressing of a button.

Risk Assessments

- **Low:** The structure of the technical risk assessment must reflect the expectations for the project while also effectively setting aside the risk aspects for the eventual milestones of the project.
- **Low:** The game decided upon by the group must also be within scope of the programmer's capability.
  - Mechanics must be limited to the three week setting, and all features must be within the programmer's skill range.
- **Low:** The process of re-learning Flash, due to the fact that Game Tech was taught a year ago, could cause complications in the programming process as the features of Flash are often somewhat limited in comparison to other languages.
  - However, Actionscript is still very much comprehensible in terms of its coding, and therefore it's only the more advanced features the programmer may want to work with that may cause conflict.
- **Low:** Developing the prototype could allocate time away from creating the technical document. However, as long as planning is set correctly, this shouldn't cause much of an issue and ample time should be available for researching flash and developing the framework for a prototype.

Milestone 1: Review

Overall, everything for the first milestone went extremely well and many extra additions were included that were not required. In terms of requirements, all project aspects were viewed and the technical risk assessment was created and uploaded onto the wiki / svn. In terms of optional requirements, a lot of research and coding was done during the first week of the project. The team met on several occasions, allowing for the programmer to easily be able to grasp the concept and begin laying out the framework while consistently presenting development to the group. A line class was created to allow for the creation of the player path through the input of points from a text document. The width of the path and size of the player can easily be altered through properly named variables. On screen, the project exhibits a simple prototype that portrays how the character would move along the path.

None of the risks were determined to cause any lingering issues that would affect the development moving forward.

Milestone 2: Risk Expectations

For this milestone, the prototype is expected which should convey the core mechanics incorporated in the game. Due to the efforts of the previous milestone, much of the foundation of the prototype has already been implemented. Therefore there are only two main aspects that are required for the prototype. First, the prototype should be further developed to allow for movement along the path to cause the player to rotate and move properly with the game's overall design. Second, the prototype should include the four basic mechanics discussed during design meetings. This second component is expected to require the most amount of time to create, as it is expected that the mechanics created in code should be near completion by the end of the second milestone to allow for polishing and further testing during the last milestone.

Required Components

- A working prototype of the gameplay should be created.
- The four basic mechanics should be implemented into the code with working gameplay.
    - 1. Dash
        - The mechanic allowing the player to dash forward by quickly tapping the button should be implemented to allow them to dart through certain objects that would otherwise destroy the player or hinder their path.
    - Fence
        - A basic fence that prevents walking, but allows dashing at a close distance should be created to place along the player's path.
    - Laser
        - Similar to the fence, however, unless the player dashes entirely over the laser, the collision with the laser should cause a 'game over'.
    - Crusher
        - A basic crusher should be created that has three parts. First, a base should be built that stands still and 'catches' the crusher face upon return. A crusher face should be created that extends outward at a specified distance and speed, returning to the base at the end of the cycle. Finally, a scaling piston should be added to reflect the extension of the face from the base. The crusher, upon colliding with the player in any manner, should initiate a 'game over'.
    - Camera
        - The player should be the center of a camera that follows the player through the level. This camera 'slides' back to the start position when the player resets to the beginning. Camera motion should be smooth (no stuttering).
    - Guards

        - The pathing and rotation of the AI guards should be simple enough to include with the line class. However, a masking class for the flashlights / spotlights will need to be created to show transparency and the areas the player must avoid.
    - Spotlight
        - A spotlight effect allowing for a specified distance, direction, and speed should be created. Upon colliding with the player, it should initiate a 'game over'.
    - Video Camera
        - A video camera with a base mount and rotating view should be created allowing for at least a specified speed in terms of the camera swing. The camera will have a spotlight effect attached to it that, upon colliding with the player, will initiate a 'gamer over'.
    - HUD
        - A simple HUD should be created to show when the player can dash and when they cannot. The dash feature will be available on a cooldown system, and the amount of time for the refresh will be based on gameplay testing. Upon using the dash, the player must wait a set amount of time before dashing again, and the purpose of the HUD will be to at least show when the dash is available. Later development may allow for a transition to better convey when the dash is about to be off cooldown.

Optional Components

- The basic requirements for the game will simply require a working prototype and the four mechanics to show the simple gameplay, however further work can involve polishing the gameplay, including art, improving designs.
- The creation of menu screens / game state screens, can be implemented.
- 3. Bar System
    - Since the player will sometimes be required to unlock doors or disarm alarms, a bar system must be implemented that will allow the player to tap the button when the bar is aligned with the indicated stopping point. Such a system should show an animation of an oscillating bar and convey failure / completion depending on their performance.
- 4. Turret
    - A rotating enemy should be created that will fire projectiles at the player if within vision. Further AI may be developed later, but for now if the turret's rotation and sight is on the player it should simply fire a harmful projectile.

Risk Assessments

- **Medium:** Dividing time to tackle the progress of the mechanics will require planning and research that should be viewed immediately after the basic functionality of the line path and player movement is cleared. Therefore, the first aspects of the prototype must also be complete as soon as possible to make room for the development of the mechanics.

- **Medium:** The creation of the masking process for the spotlight may cause some time delays. The programmer has done them before in flash, and therefore has sources to reference in order to achieve objectives, but the process can still be difficult, especially with multiple lighting effects occurring. Therefore, after the guard movement is implemented, a large amount of time should be predicted for the development of the spotlights / flashlight visions and player collision.
- **Low:** The Dash mechanic should be relatively simple, the only known bug that could limit time occurs if the player attempts to dash around corners, but due to the state and flexibility of the current Line class, this should not be a problem as the player move through interpolation.
- **Low:** Screen and HUD mechanics should be easy, and since they're optional, they shouldn't be an issue anyways. They'll be developed at some point, and can be focused more on during the last milestone. However, neither can be tackled until the designers and artists have their parts, so that should at least be acknowledged during milestone 2 in order to prepare for the development of the last stages of the game.
- **Medium:** The amount of instances required for the spotlight (spotlight, guard, and video camera) could take some time in order to properly build with substantial polish. It may not be possible to complete all features of the spotlight, but at the very minimum, it should be expected that the spotlight alone should be implemented with collision.
- **Medium:** The crusher, while seemingly easy, could cause some issues upon attempting to sync all three components. In other words, combining the piston, face, and base may cause visual problems depending on the distance or speed specified in the text file. Use caution when building the crusher system, as it must be implemented as bug-free as possible for best effect.

Milestone 2: Review

One of the main aspects to recognize at the end of Milestone 2 is the amount of design alterations that occurred in the first few team meetings. For instance, while a bar system was previously mentioned in the first milestone and left as an optional feature during the week of Milestone 2, it is likely that the entire feature may be left out in order to focus on other aspects such as polish, checkpoints, game screens, details, and animations. Furthermore, the turret system is still planned for the final week, and the guard movement may still need some further additions in order to have enough quality to be presented for Alpha testing. What should also be noted is the reality that in the current testing process of the game, the absence of the bar system is not noticeable, and furthermore, gameplay focuses more on movement, timing, and strategy. So, at the current moment, the addition of a bar system would halt the gameplay with a game mechanic that varies drastically from the other pieces, and only with precise design implementation would the feature actually work. So, the bar system will likely be left out for the final week.

Overall, Milestone 2 went largely according to plan, as the majority of the desired features were not only implemented without any apparent bugs, but they were also designed in such a way that the designer on the team should have little problem testing the game with text files to create objects that can have a set location, rotation, scale, speed, and distance. This offers a lot of potential for level design, and although a complete level editor would be extremely useful for further development, given the three week time restraints, the text file procedure offers more than enough accessibility and allows the programmer to focus more on the perfection of the implementations and class creations.

All of the mechanics that were implemented were not only successfully built, but there was more than enough time to effectively debug them in most instances. Dashing works properly with all the mechanics presented currently. Fences stop the player movement and allow dashing to pass only if the player will get over the fence at the end of the dash. Lasers exhibit collision if the player walks into them or lands on them after the dash. The spotlight effects (spotlight and video camera), rotate back and forth properly and collide with the player. The crusher, while taking a lengthier time to code, has three synchronized graphics that move at a speed and distance that can be specified by the designer. The designer can easily add a line to the crusher text file to convey the location, rotation, scale, speed, and release distance for any number of crushers. This task took a lot longer than expected simply due to the complications that occurred when attempting to scale the piston, as the process required for the piston to extend in one direction with the crusher face without extending out too little or too far. Since the program allows for any rotation, the movement in both the x and y planes had to be taken into account when the scaling of

the piston. The camera following the player was implemented properly with smooth transitions, as it is not only able to reset to the player's location at any moment, but it also slides nicely throughout the available map. Finally, the HUD was easily added to show the cooldown timer, and it should get further animations from the artists in the future to properly convey when the dash is about to be off cooldown.

The only aspect of the required components that was not fully completed was the guard movement. This requires a guard following a set path while carrying a flashlight that acts in a similar manner as the spotlight and video camera. Originally, the flashlight effect was going to be difficult due to the expected masking process, however, by simply using a transparent 'cone' or 'circle' with a tint of yellow, it will possible to render a similar effect that can be improved by the artists at a later time. This aspect was already used to create the spotlight and video camera. Still, the guard mechanic was not completed due to the fact that other mechanics, such as the crusher and fence, took a bit longer to debug after creation. It may be possible to continue working with the mechanic before the prototype is expected, but depending on how much is built, it may not be desirable to present the mechanic until the next milestone. Still, the guard mechanic may be achievable in the near future as it is likely that the guard and turret mechanic are the only unknown factors required for the third milestone. Aspects such as checkpoints, game screens, and polishing should not require external research or excessive testing.

In conclusion for this review, it should also be recalled that various art assets were implemented into the game. However, much of the art can still be considered to be at the 'prototype' stage as much of it requires proper scaling and additional animations. Still, the majority of the art fits within the game world and combines well enough with the rest. Overall, this milestone has been completed close to expectations, with the exception being the guard mechanic that may not fully be ready by the prototype session. Still, with the understanding of the amount of flexibility that was given for all the other mechanics currently in play, the designer should have more than enough during the third week to begin crafting levels while the final two mechanics are built and ready for polishing stages as this project approaches alpha.

Milestone 3: Risk Expectations

For this milestone, all of the core mechanics of the game should be in place, along with a few sample levels to reflect the game's Alpha status. Art, when available, should also be on the primary list of objectives in order to further convey how the actual gameplay would appear in the beta stage. At the end of this milestone, the design for implementing additional levels, displaying win screens, keeping track of checkpoints, and monitoring the player's progress in each level should be obtained and finalized to make room for polishing in the beta stage. As a result, the final touches to mechanics should be added, checkpoints should be created, tutorial text should be implemented, game screens should be displayed to the player on loss / level complete, and art should be combined with the game whenever available. Furthermore, the game should run as a .swf by the end of this milestone.

Required Components

- Turret Mechanic
    - The turret mechanic should be implemented with art and rotation. The turret mechanic will involve a rotating visual with a laser detector. When the laser collides with the player, the turret should fire a rotating projectile and wait a set time before continuing its rotation cycle.
- Guard Mechanic
    - Although primarily implemented during the previous milestone to further show the prototype gameplay, the guard mechanic should be reviewed in order to allow for guards to rotate at each point, giving their flashlights more potential to collide with the player.
- Checkpoints
    - In some manner, likely via specifying points, the checkpoint system should be implemented such that, in each level, the player will respawn at the nearest checkpoint that they have already gone past. While seemingly simple, checkpoints must take into account the process of respawning on the line system.
- Tutorial
    - Although slightly unclear in terms of aesthetics and design, the tutorial level should be designed in a manner that has enough checkpoints and simple mechanics to convey the basic gameplay functions. Furthermore, some system should be in place to instruct the player how to proceed. This will likely be done with signs and text currently.
- Animations
    - Once art is available, the Cat and Dog sprites should be animated in accordance to their actions.
- Game Screens
    - Once available, although placeholder art can be utilized in the meantime, screens should appear at the end of each level, at each game over state, and at the start of the game.

Optional Components

- Wall System

    - Although not required for Alpha, the actual system of placing visual objects could be implemented to further display hallways and other objects to create the experience on the interior of the facility.
- End Game Visual
    - Simple to program, but if available, an 'reward' graphic should be placed at the end of each line to convey the reason the player is going through the level. This waits largely on a visual or design decision.

Risk Assessments

- **Low:** The turret should cause any real issues, as the rotating and scaling processes have already been done numerous times throughout the project. Still, this might as well be noted in the case of any design decisions with regards to the turret's function.
- **Medium:** Rotating the guards could require a lot of tinkering with the movement feature at there is no simple way of halting the progress of guards on the lines. Guards currently do not stop at each point, but instead they realistically walk through all points in accordance to their speeds. Therefore, some trigger will have to be defined to detect when they have reached the point, and then their progress must be halted until some rotation sequence completes. The rotation sequence will also have to be in the shortest rotation possible, which will require further calculations.
- **Medium:** Checkpoints are only any issue due to the research and thought that will be required in order to figure out how they'll work. It is simple enough to have the player respawn along points, but the goal of check points is to allow the player to theoretically respawn anywhere along the line. Manually inputting points into a text file would not work, since on diagonal lines it is much more difficult to easily figure out where to spawn the player. This will require some planning.
- **Medium:** The tutorial will only cause some issues as it will require the programming to essentially figure out how to explain the game's features without additional assistance from other roles. Display text on signs is simple, but may not fit with the art style or make sense with the design. This will require some strategy to tackle.

Milestone 3: Review

Milestone 3, as expected, went mostly according to plan, as most of the aspects cut only impacted other roles and did not primarily influence the programming component of the project. All required features were implemented properly, and some of the optional features were added as well. The system in place will allow for easy additions of design and art, as all of the mechanics are in place and the level creation process is set to allow for further levels in cooperation with the win / loss screens. Overall, milestone 3 likely took less time than milestone 2 due to the smaller amount of tasks.

Turrets rotating and fire as expected, and some research was attempted to give turrets a limited rotation range on if desired, but the original system was left in place due to complications during the research. The guards rotate correctly at given speeds, and the only slight issue is the occasional visual glitch that occurs. Bug-solving in these areas was sacrificed in favor of allowing for further time in terms of level creation. The checkpoint system, surprisingly, went much better than expect. The designer would not have to state actual points, but rather, in each line of the checkpoint file, they can simply note which section of the path to examine and what percentage along that path to have the player respawn at. Then, during gameplay, the correct checkpoint will trigger behind the player when they respawn. This implementation went far better than originally expected.

The tutorial also went much better than expected, as designer and art assistance from Alexander Beauchesne during research allowed for a much stronger instruction method. Players will now approach tutorial bots during the tutorial level, and upon reaching a certain range, the bots will display visual guides for various aspects of the gameplay. While requiring some further perfection in the future, the tutorial art aspects in place from Alex will allow for simple visual instructions to guide the player through some of the more prominent mechanics in the initial stage. For the tutorial stage, the prototype, already meant for the tutorial aspect, was simply adjusted in order to add checkpoints and other minor mechanics to guide the player along the initial steps of the game.

The game screen and animations took some fiddling, but ultimately they were simple to add onto the game and only required sprite sheets or images to clarify the proper scaling. The wall system was eventually discarded due to its requirement for a level editor, which could not be completed in a timely manner. The end game visual was left out for now while artists complete and polish other more important assets to be included in the alpha part of the game. Overall, the programming components of milestone 3 went smoothly in terms of adding the final parts of the project and preparing room for design intergration in the final stage.