CS4900 Capstone Project Proposal Document

Group Members and Group Number:
Emily Shirley, Carson Davis, Mathieu Davidson; Group 3

Project Title:
Tacticool

Project Executive Summary:
We are building a browser game with Javascript and ThreeJS, a 3D graphics library. Our genre is a fusion of a Tactical Role Playing Game and a Rouge-like. The mixture of these genres as well as 3D graphics offer an interesting set of challenges for the user that we are excited to create and share with our players.

Problem Definition:
Strategic Role Playing games (RPGs) tend to have multiple layers of depth. So much in fact, that playing one often consumes a large amount of time. Rouge-like (RL) games on the other hand tend to offer short bursts of gameplay. These "runs" can be enriched by varying levels of randomness. We wanted to create a game that appeals to both those who don't have much time on their hands as well as those who can only play for 30 minutes a day.

Tacticool will contain mechanics that involve attacking and being attacked by others, so it would receive a rating of E10+ for "mild violence". As the rating suggests, our target market would therefore be for ages 10 and up. As for the gender of the target market, RPGs such as *Tacticool* can appeal to any gender. Since it is a web-based game, the user must also own or have access to a computer with an Internet browser, however, our proposed game will only be single-player so users will not be able to play with their friends. Lastly, the organization producing this game consists of a small team of Computer Science seniors from Valdosta State University.

Proposed Solution:
We are applying the techniques and practices we have learned during our time in the Computer Science program at Valdosta State University. From implementing Software Engineering principles to analyzing the benefits and setbacks of data structures in our build, this project involves all aspects of our education. The freedom to use our acquired knowledge to bring our ideas to life also had its pitfalls, however. With this freedom comes a great responsibility to limit our goals in a way that makes the project feasible within the given timeframe without leaving out key elements that make it stand out. Doing so is a great skill that will benefit us as software developers throughout our careers.

As previously stated, we seek to create a solution to the problem that many Tactical RPG players face: the games usually take far too much time to complete. In addition, we wanted to not only appeal to those who lacked time but also those who have more time to play games. Our solution is to create a game that is strategy-based but also, quick, random, and unforgiving. This is where *Tacticool* comes into play. Tacticool will apply the quick and random nature of RLs with simple Tactical RPG mechanics to provide a go-to game for those who enjoy strategy but also desire faster and less forgiving combat. Due to the random nature of our game, the replayability is increased, allowing for extended periods of exciting, non-stop gameplay. Our game will encompass turn-based combat in order to satisfy the RPG aspects that we want to include. However, it will also have randomly generated levels that increase unpredictability, while maintaining the RPG aesthetic. When a level is randomly generated, the terrain, structures, and enemy placement will all be randomized in a way that the level is still beatable. To do so, we will need to utilize algorithms that can only place enemies or obstacles in certain positions on the map. Additionally, once a character is defeated, that character can no longer come into play. This permadeath feature of the game helps us involve the RL characteristics that we believe are necessary for a quick and replayable combat game.

We have been holding weekly meetings during which we discuss the project and delegate tasks. We have found that self assigning separate jobs has been the best way to ensure we are maximizing throughput while maintaining a cohesive direction. Once we have our assignment for the week, we typically must spend some amount of time researching possible techniques to implement. Much of this research is performed independently. Occasionally, when one of us reaches a branching point (i.e. choosing one method or library over another), we will have a mid week check-in similar to a stand-up meeting. During this time we poll each other about the choices that must be made, make decisions and begin working. During the next phase of development, we intend to complete basic mechanics. Following that, we will design tutorial levels to incorporate into the game. While implementing our designs, we use various tools and software to bring our vision to life.

We are building this game with the intention of running it in web browsers, particularly Google Chrome. In order to render our 3D graphics, we are using the ThreeJS library. ThreeJS was built on top of WebGL (Web Graphics Library) which allows GPU-accelerated usage of physics, image processing, and effects as part of the web page canvas. With ThreeJS, we are able to gracefully manipulate visuals with programming. Another useful tool that this project will utilize is Yuka. Yuka is a JavaScript library that contains numerous AI movement implementations which we will use for the enemy team within our game for things such as following the player's character until they are in range or staying away from a potentially dangerous situation. The characters themselves use free models provided by Quenternius, and can be found on his website (http://quaternius.com). His site has numerous models to choose from, and they all come with animations as well.

Regarding the programming languages used within this project, the main bulk of our code is written in JavaScript. In addition to the backend JavaScript code, we also incorporate HTML and CSS in order to create appealing buttons on the Heads-Up Display (HUD).

In order to work collaboratively, we are using Git for version control and hosting our code base on GitHub. This allows us to contribute remotely while minimizing conflicts within our work. Another asset that we've used frequently is Google Docs. We use Google Docs to collaborate on project documentation together in real time. Google also provides Google Sheets which we have used to collaboratively record our project log that contains the various tasks we've performed and what dates they were performed on. The Integrated Development Environment (IDE) that each of us is using to work on this project is Visual Studio Code, and we use Discord as a form of communication to schedule meetings and discuss the project. While we are not getting involved with the creation of models for this project, we have been slightly editing the models we've acquired. To do this, we are using a software called Blender which is a free 3D computer graphics software. Lastly, each team member has his/her own laptop that has the software and various utilities needed to contribute to the project.

To complete the creation of *Tacticool*, we aim to hold meetings at least once a week to solidify our roles for the upcoming week as well as discuss the project as a whole. These meetings also provide us a way to discuss issues that we've come across or request feedback from other group members. In the upcoming weeks, we aim to solidify the base elements of the game, resulting in a working product. As we build upon the game's functionality, we will be creating a static tutorial level that introduces the player to the game's mechanics. Additionally, while this progress is being made, attention will also shift to the title screen creation. We will put thought into polishing up the game by creating various level designs, as well as adding items and abilities to each character. Lastly, we will focus on the more creative aspects of the game such as music, sound effects, and model colors.

We aim for the budget of this game to be $0 since a plethora of free software and models can be found and downloaded online. The software used to build this game is all free to use, and the models we acquired are free and under Public Domain.

Timeline:

We have 8 weeks left to deliver a finished product. Already having much of the preliminary work finished, I submit that it will ideally take 3-4 weeks to integrate our work into a solid alpha build. This leaves the remaining 4-5 weeks for improvements and polishing. To achieve this build, we plan to accomplish the following: a title screen, a tutorial level, some form of enemy AI (this includes both movement and attacking), and finalizing our terrain generation to include spawning for characters and obstacles ready to be utilized by both the player and the enemy. Once these integral functionalities have been introduced in some reliable manner, we then want to work on introducing additional gameplay details. These will include additional random level generation, support items, class abilities, and music / sound effects. A rough timeline that we currently have in place is as follows:

Task	Tentative Completion Date
Title screen and functionality	3/23/2020
Tutorial level design	3/30/2020
Tutorial level creation	4/6/2020
Enemy AI implementation	3/23/2020
Terrain / Model relationship	3/23/2020
Terrain object spawning	3/23/2020
Item Creation	4/6/2020
Additional level types (random level generation)	4/9/2020
Ability Creation	4/13/2020
Music / Sound effects	4/20/2020
Last minute touch-ups	Week of April 20th

Reflection:

This project has proven to be both formidable and rewarding. It has allowed us to exercise our discipline with no restraints on our creativity. As such, it is easy to let ambitions outgrow reality. One must be aware of the calendar as well as the scope of the project at all times. This awareness translates to time management as well as overall project management. Keeping up with work completed through a shared Google Sheet as well as posting tasks on our GitHub project board has helped with managing workflow. With this project having a considerably longer timeframe than we are accustomed to as students, it is easy to feel complacent regarding the deadline. Tasks can stretch on longer than we expect and grind production to a halt, making efficiency a high priority. It is highly important to recognize when a method of performing a task should be changed because the time to implement it would be greater than the amount of time that should be dedicated to it.

Utilizing languages and code libraries that we are not entirely familiar with has also been challenging. The asynchronous nature of web programming affects the way assets load, which can be problematic when manipulating them. Additionally, it is easy to accidentally create inefficient scenarios if the code is not optimized. One of the largest aspects of this capstone style project is the necessity of researching technologies that are both new to us individually, and new to the world of 3D, web based, game development. One of these aforementioned new technologies is ThreeJS, which is the fulcrum of this project. None of us have had much in-depth experience in JavaScript before, so this 3D graphics library has been our latest, and potentially greatest, hurdle regarding the technology we are using. Another tool we currently have in our disposal is a JavaScript AI library made explicitly for ThreeJS called Yuka. Despite not currently being used by us, it is a tool we are already planning to integrate into our game within the near future. Any other technologies we currently have in use are just extensions of ThreeJS directly and these include: OrbitControls, GLTF Loader, Object Loader, Trackball controls, Vector3, and Plane Geometry. As we continue to move forward, there may be more technologies that we find useful towards our end product and as such, we will implement those as well.

In conclusion, this project has been an immense learning experience for our team. The asynchronous nature of JavaScript has given us a new way to think about piecing together code. It takes great skill to successfully create a project as large as this one in a seemingly uncontrollable environment, and we have grown our programming skillset in doing so. This project not only needs coding skills, but project management skills as well. With any project this size, time management and task delegation are extraordinarily important in keeping the team and project on track. The creation of *Tacticool* has given us great practice in both of these aspects, reflected by our weekly meetings and continuous communication. Learning solo is not for everyone, but coding with a new language or framework requires individual research skills that are invaluable. These skills could be applied in a future job as well; while some jobs offer in-house training, most programming jobs expect you to figure things out on your own. The knowledge and various experiences we have gained from this project can be directly applied to such a case. The knowledge, understanding, and skills we have gained regarding programming as well as team management are a substantial first step in becoming professional software engineers.