

Stand firm

Game Development Post Mortem



last update: apr.10, 2022

University of utah eae 2100

u1351288@umail.com

Github: https://github.com/DefenderJohn/StandStill-Game-Project

## What is done correctly

### The story design

The basic story of this game is not just randomly thrown there. Since I’ve already got a worldview of my other game, it should be a wise idea to put this game into the same universe. To make it more clear, this game has a background story on WWII, as a commander of the tank company that stands still under enemies’ attacking, which will have some correlations with my other game.

Even if we ignore the universe design (since it’s still not yet done), the story itself is simple enough to understand, while still keeping lots of detail that can be treated as an optional side story to add to the game. This enables me to design the game in a more flexible way that as long as I stay on the fundamental story timeline, there will have nearly no chance for me to drive off the track.

It’s not yet to make any conclustions to this game because it’s still in a very early stage of design, but I believe the story part will end up well.

### The suppliment system

The suppliment system is a worth-attempting feature that will bring huge change to player’s strategy making. The suppliment system is very rarely mentioned in mordern games, and part of the reason is that we don’t know how to make it interesting in a FPS or AVG game. I won’t disagree with those big companies’ decision, because this is, indeed, boring if implemented into a first person perspective game.

If we look at the traditional tower defence game, the suppliment is sometimes been treated as healing functionality implemented by putting some “healing tower” next to someone, or the entire suppliment system is been ignored because they consider it’s “lack of interesting”.

However, in this game, it will bring a huge difference because player is no longer controlling any single soilder (or tank), instead, they are controlling a more complex team that working together to achieve the goal. In this case, we are not abstracting anything into pure data combating but actually enable player to have control on the entire frontline. In this case, if player performs well on the suppliment system, they may not need to fight as hard but can still achieve the goal. In another word, this will bring the player a new perspective, or a more realistic perspective to view how the war field actually is. I’m not saying this is a simulator, because this is a game, game should have some level of abstract so that player will be able to control the entire war zone by only himself, but having a more realistic perspective is always good when it’s not causing the lose of interesting.

The game implements the suppliment system as supply line and air-drop, player can enable the supply line to supply the tanks in front, and they need to figure out a way to keep the air-drop resources under their control. This brings more challenges and more short term goal to the game, which I believe should be good.

## What is done Incorrectly

### The goal is too large

This is a very real problem. The game itself is good, but there is not enough time for me to complete the entire game.

Although I treated this as a long-term maintained game, but there still exists something that I must done in this semester, which causes huge trouble to the entire process.

In the very early stage of development, I was considering the difficulties of designing the navigation system and implementing tanks should not be that high, or at least it will not cause me unable to finish at the end of the semester. However, I should say that I made an optimistic assumption. The technique difficulty starts existing from the very beginning of the development, I was using unity version 2022.x, however, there exists a huge error when I’m attempting to implement several FBX models.

This error is really interrupting the entire schedule because, under the consideration of project safety, I had to re-roll all the project content back to a more stable version, which is 2019.x, but it also means that I’ll lose all my work because there is no way to safely transfer the same functionality from a later version to an earlier version. This forces me to delete the entire old project and start over again, which loses me huge amounts of time.

Besides that, there exists some script APIs that are only available in the newer version, which means I have to either give up the attempt to implement the functionality, or I need to write the code to implement the entire API by myself. I’m still making attempts on trying to get all the scripts done, but it shows that it’s driving further and further from my original schedule.

In order to make sure that there exists an at least playable game there, I must choose to throw out some functionality and focus on the more important parts. This is not some strategy I would like to take, but I have to.

The thing I can learn from is to always plan in a most conservative way so that even if something really unexpected happens, there still exists some room for one to maneuver. Otherwise, if pushing oneself too hard in the first place, then the result may not end up well if any exception shows up.

### Rely on external resources

It’s very common in game development, but it’s not something pleasant to find external art resources. During the version switch, some of my old art resources from the unity asset store are no longer available to use, but the core functionality is already there, which means I have to either find some really similar external resources to reduce my working load, or I have to make a re-design to some basic functionalities or even the game feature—you simply can’t build up a tank game without even having a tank model!

It’s lucky that I found something that can be treated as a replacement for the original resources, but the downside is that I’ve already spent too much time searching for external resources that I have to compress the time to actually implement all the functionalities. This is also one thing we need to learn from.

### Falsy estimation of code scale

This is more a practicing issue on code style. In the very beginning, I do not estimate there will have too much huge structure to design, so I keep the code style in a most trivial way by just throwing different functionalities into scripts and throwing scripts onto the actor I want it to apply.

This works when I’m in the very early stages because there haven't been added too many things and everything seems under control.

However, as soon as I entered a more complex stage of design, the badly designed code style soon becomes a major problem. When I want to add some functionality, I need to change the code in several different scripts and remember them in mind. After several hours of struggling with slow work, I decided to completely rewrite the code structure so that it will keep the code readable and healthy for maintenance. The new code style is better for larger projects, which I should have used in the very first time establishing the project.

The re-write and the slow work are all due to the falsy estimation of the code scale, I should learn from it and make sure next time I always start by estimating the project in a most aggressive way and reduce the chance of losing control of the code structure and styles.