# CS 3540 – Gold Master Release

In this deliverable, you will build on your beta release by adding more functionality to it and produce the final release of your game. This is the final release of your game, which will be used for final grading. Essentially, this is your “Gold Master” release that would have been sent for master production if this were a real-life game development project. You will also submit a project report, along with your updated game design document. Here are the requirements from the Project description (posted in Week 1). Your final game should

* Show that you did your best to produce the best game your team can.
* Be a highly polished 3D video game.
* Feature high interactivity and appropriate visual and audio effects.
* Use at least three different animations for visual effects.
* Be highly polished and never use primitive shapes or default Unity skybox (solid blue) as the only assets.
* Have a clear story and should support progression.
* Have at least three levels with various game mechanics in each level. Each level should feature at least one new mechanic/loot/particle effect/interaction technique/etc.
* Have an intuitive user interface that displays progress/time/health/damage/etc.
* Use a game manager that keeps track of time/level/health/ammo/etc.
* Implement game AI and use intelligent behaviors for enemies and NPCs.
* Have a main menu that allows the players to start the game and keep track of their progress. The main menu could, for instance, display the amount of time the player has spent playing the game. The main menu should also include a section that displays information about your team (game programmers). Player should be able to change at least one game setting (e.g., mouse sensitivity, player avatar, etc.) using your menu.

### NPC Requirements

1. NPC AI should use an FSM with at least two states (Chase and Attack). The FSM should be similar to the in-class example. If your NPC can support a Patrol state, that should also be implemented. For the Patrol state, the NPC should wander around some points in the game. That said, not all games are amenable to implementing a Patrol state.
2. When the NPC attacks the player, it should cause some damage to the player/target. Attacking may have a totally different meaning in your game (e.g., waving and saying “Hi” when the player is close).
3. Implement a sense of **sight** for your NPC. These senses should be integrated into the FSM so that the NPC starts chasing the player when it can see the player based on the sense of sight you implement. Depending on your game design, this functionality may be omitted, if the NPCs are designed to directly attack the player/target.
4. The movement of the NPC should be based on the NavMesh system. For this purpose, you will need to bake the NavMesh first. Then control the movement of the NPC using the NavMesh system.
5. Please note that this NPC doesn’t necessarily need to be an enemy. A game agent could be pacing on its own normally, but when the player approaches it, the agent could change its animation to talking and share a tip or level goal. There are many ways to implement intelligent behaviors for NPCs, and we would like you to try your best to include one in your game. If you cannot come up with an idea, implement a tutorial agent that introduces your game to players when they first play the game. This could be a different scene of your game. Instead of text instructions, you can use an online text-to-speech tool to have your agent speak the instructions (by saving them as audio clips and then playing them in Unity when needed).

### PLAYER REQUIREMENTS

1. The user should be able to control the player with no problem. Camera view and player movement should be highly functional with no obvious lags or issues.
2. Appropriate game mechanics should be implemented for your player. This is going to vary greatly depending on your game, so be reasonable. At the very least, we should be able to move or change the perspective of the player and jump.

### GAME POLISH REQUIREMENTS

The overall polish of your game is important as well. 20% percent of your grade will come from how polished your game is. Game polish refers to the look and feel of the game and reflects how much effort you’ve put into making the best game possible given the requirements. For the most part, this is a subjective quality of your game; you’ll know it when you see it. As a rule of thumb, please make sure your game is not half-baked; there should be no primitive shapes without materials, for instance. Everything in your game should indicate that you’ve spent considerable time fine-tuning the details. Another example is related to the UI. All UI elements should be reasonably sized/aligned (e.g., no random alignment of score texts). Even using a custom font typeface helps with game polish. Be reasonable, and you will probably be fine. Ask yourself if you would be comfortable featuring this game on your portfolio and sharing it with potential employers. That’s what game polish means!

Here is the scale that will be used to assess the polish of your game:

**Excellent:** presents a highly polished game release; goes above and beyond the requirements to improve game mechanics or to add new ones; provides aesthetically pleasing look; adds extras when applicable to make the game interesting/innovative; uses realistic behaviors and appropriate graphics instead of simple shapes.

**Satisfactory:** uses the same game mechanics introduced in class; meets the requirements; graphics look good but can be more polished; no extra effort to make the most interesting/innovative/different game; uses simple shapes sometimes to meet the requirements

**Half-baked:** one or more requirements is missing; problematic game mechanics/weird controls/behaviors; graphics/colors/assets don’t look very good; no extras; gets away with simple shapes for the most part; doesn’t convey an effort to produce the best release possible

**Dull:** bare minimums in all aspects; multiple problems with mechanics/graphics/assets. No effort to make the game look and feel good at all.

**Please delete the first page AND all the prompts. Submit a well-formatted document. You will lose points due to formatting issues (up to 20%). Please follow the instructions!**

# CS 3540 – Gold Master Release <Insert Game Name here>

## GAME DESCRIPTION

Design a cover image for your game. This would be the feature image displayed for your game; it is like a banner of your game. Some teams simply use the intro page of the game with the name of the game displayed. Some get more innovative and come up with something cool. It is up to you but do your best! Refer to Steam games for inspiration. 😊

Provide a description of your game in 50-100 words. The idea is to capture what your game is about and pique the interest of potential players. This would be the game description displayed on Steam if your game were to be released there.

## WEBGL EXPORT

Provide a link to the WebGL export of your game that is **public** and playable online at Itch.io/Unity Cloud/Github/etc.

## REPOSITORY

For your final release, we also need the entire Unity project (with all the assets, files, scripts, etc.) Therefore, you will need to upload it to an online repository, such as GitHub, where we can download it. It is very likely that at least one member of your team has a GitHub account. 😊 Other repositories, such as Google Drive, are fine too.

Make sure to include the **public** link to your repository here.

## GAME ASSETS

How did you find your assets? Provide links to the resources you’ve used.

## GAME DESIGN DOCUMENT

It is likely that your game design has changed drastically since you submitted your initial GDD. Based on your final game release, provide your updated GDD below. Remember to **delete** all the prompts!

### Characters

Who are the main characters in your game?

*The reason we start with characters is because you need to introduce them before the Story.*

*An example of character descriptions:*

***“Gnumies*** *are the main characters in this game. These creatures are happy and wealthy, but not greedy. They are wealthy because their ancestry is related to money, or Numismatic, thus their name: Gnumies. They’re hairy and come in a variety of colors.*

***Red Gnumies*** *are passionate and break stuff.* ***Yellow Gnumies*** *are electric and jump up and down.* ***Green Gnumies*** *are tranquil, relaxed and easy going.* ***Blue Gnumies*** *are a little sad and grumpy.*

*Gnumies also have a lot of arms, anywhere from 1 up to 4, and their arms have hands. They have a firm handshake and can combine when holding hands. Gnumies like rough play and leave everything messy…”*

*You can also add some character artwork here, if any.*

### Story/Narrative

Having introduced the characters, it’s a good time to talk about the events that will happen throughout the game. What is the story/narrative of your game?

*For example:*

*Gnumies are happily playing inside their castle and causing mischief. The Butler is going insane, but everybody is enjoying. Joker makes jokes.*

*German is home watching TV and his mother bothers him. So he goes out to spy on the Gnumies. Outside is raining and German is looking envious through the window.*

*A strange mysterious person gives him a key that he can use to enter through a backdoor. He goes in with his army, kidnaps and jails female and baby Gnumies, and kicks everybody else out of the island…*

### Game World

Describe your game world/environment. Where does it take place? (dungeon, space, sci-fi town, etc.)

### Gameplay

How does the gameplay work? What is the player’s goal? Why is the player playing your game? Is it possible to classify this game into one of the common genres?

*For example:*

*Overall (long term): Help Gnumies return home*

*Gameplay (short term): Defeat the enemies, advance to the next level…*

### Game Mechanics

Describe the core game mechanics, using verbs. Remember these are the methods agents can invoke to interact with the game world. Include subsections for players, enemies, and non-player characters (if any).

### Items, Loots, and Power-ups

Use this section to elaborate on the Game Mechanics and talk about things that can be added to the game in order to improve the fun and empower the player. In this section you’d add every power up and item the player can use/encounter/buy and how they would affect the core gameplay.

*For example:*

*When finishing a world, you could get a power up related to that world. For example, finishing the volcano world, can give you an item that makes red Gnumies more powerful. It could be a scarf, or something they can wear, and those items could be seen in-game later. You can level up items using in-game currency, or use real money to acquire in-game currency packs…*

### Game Rules

Bringing everything together, what are some game rules for winning, losing, beating levels, health, and damage?

*For example:*

*These are the losing conditions: losing by running out of time, losing by running out of moves, losing when there are no available combinations.*

*When the player loses, there must be an image showing the Gnumies wounded/scratched. Maybe they can lose some hair and you can see the skin under the hair.*

### Target Audience

Who is this game for?

### Artwork

What artwork did you need? Did you search online? Did you use any of the assets provided in this class?

## PLAYTESTING FEEDBACK

Similar to the alpha release, playtest your beta release with friends and family members. Each member of your team should test it with at least one player (there will be an OLA about this in Week 14).

Explain how you incorporated the feedback from the alpha and beta playtesting sessions into your final game release. In what ways were they useful? What did you add/remove/change/keep?

## CONTRIBUTIONS

In the table below, add the name of each script/game component. and the names of teammates who have contributed to that script, along with a brief explanation of who did what. Two sample records are provided.

|  |  |  |
| --- | --- | --- |
| **Script/Component** | **Contributors** | **Description** |
| MouseLook.cs | Harry & Hermione | Harry: FunctionA()  Hermione: FunctionB() |
| Main Menu Design | Ron & Hermione | Ron: designed the main menu  Hermione: implemented the script to make the menu functional |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**WHAT TO SUBMIT**

You should complete this document, delete all the prompts, and submit a well-formatted final report (PDF file).