Cpt_S 487 Project Specification

1. **Project Name:** BH-STG: Bullet Hell Shooting Game & Level Interpreter

Final project is an important component in a software design class. You will complete the project in a team environment (each team will have max 4, min 3 students, with 5 or 2-person team per instructor's permission or recommendation). The project is to provide a platform on which you can exercise software design and decision making on software architecture.

For this semester, your team project will be to implement a bullet hell shooting game, and its level interpreter, inspired by an indie Japanese doujin game series: Touhou Project. A sample game play could be found here: https://www.youtube.com/watch?v=-tyPdlhMLFQ
The genre mostly originated from the arcade games, notable examples include

- Touhou Project (http://en.wikipedia.org/wiki/Touhou),
- Galaxian (http://en.wikipedia.org/wiki/Galaxian, https://www.youtube.com/watch?v=XhYVcwhSWjI)
- Raiden: (http://en.wikipedia.org/wiki/Raiden_%28video_game%29, https://www.youtube.com/watch?v=Xe_5zxXiQAs)

Bullet hell is also a key element in one of the most popular recent games: Undertale.

The basic concept of such games is quite simple: the player controls a character/spaceship, which shoots bullets/lasers at enemies to advance the game. Meanwhile, the player has to move and dodge a "large" number of enemies and the projectiles they fire. These projectiles sometimes form certain patterns to increase the difficulty and/or aesthetics of the game. At the end of each stage (or sometimes in the middle), the player will fight a "boss" who is usually more powerful and have more elaborated attacks. Other mechanisms (bomb, scores, power ups, etc.) are often available in such games as well.

The end goal for the project is to design and implement such a stand-alone desktop game and its corresponding level interpreter. There are three major deliverables for this project throughout the semester. This document explains the scope of this project, what is required for each deliverable, and other necessary details. Note that some requirements of the deliverables are intentionally withheld temporarily in this document. This is to encourage you to really think long and hard about your design before coding, and so that you might experience the challenges when facing changing requirements. These additional details will be announced in the "**Course Announcements**" forum once the previous deliverable is due.

2. Overall Project Instructions

1. **All codes should be submitted to github.eecs.wsu.edu**. Please see Assignment 1 for details on setting up your EECS git account. Your team should collaborate via git for team coding as well. DO NOT make the following activities regular occurrences within your team: email/message your codes back and forth; using a flash drive to carry the code; executable only on one team member's computer and that computer only; other similar methods.
   You should make learning git one of your main goals for this course. Occasional offline "copy and pasting" is of course fine, but don't make that into a habit.
   For the same reason, I will not accept zip files of a solution/project submitted via email or other methods. Make sure your project in your github repository is runnable (with necessary libraries installed) on a normal PC with Windows (10 is preferred) after a fresh clone, and include detailed Readme file with instructions on how to run the program.

2. **You should use C# .NET or Java as your programming language.** Both languages work well with what we are trying to cover in this course. In particular, if you are using C#, please start researching on **XNA or MonoGame** as your underlying game engine; if you are using Java, do the same for **libgdx**. All team members should get familiar with your choice of framework as soon as possible, as your first deliverable due date will be right upon you. *Unity* is NOT allowed. Other frameworks while not forbidden, but are not recommended. Please contact me if you would like to use some other frameworks to build the game.

3. **Important:** Please note that this course is not about how to code a specific game (and definitely not about how to play the game). Therefore, we will NOT cover any specific engine-related contents AT ALL, which means you are responsible for using all resources you can find to learn about **XNA/MonoGame** and/or **libgdx**, and use them effectively to complete your project assignments. I strongly encourage using the discussion forums for such purposes if you need help and/or want to share wisdoms while learning the frameworks.

4. **Crucial: Read carefully and ask questions!** In order to include as many details as possible, this specification is terribly verbose, which will likely lead to confusions and even contain unintentional mistakes. Therefore, please be sure you read the requirements very carefully for each deliverable, and post any questions you might have on the "**Questions for Instructor**" forum. Remember, good questions count towards your constructive posts!

5. **Demos:** Throughout the semesters, I will try to schedule team meetings (via skype or even AMS video conferencing) with each team every few weeks to check up on your progress, answer all your questions, get your feedback and keep in touch with all of you personally. These meetings will be announced in the "**Course Announcements**". The dead week will be used as the project demo week, including the weekend. Each team shall schedule a final demo with me during the week with everybody attending. You will demo everything you have done to me and explain your contribution.

6. **Grades:** details are included below, but the overall distribution of your project grade is as follows:
   a) Deliverable 1: 20pts (+up to 10 extra pts)
   b) Deliverable 2: 40pts (+up to 5 extra pts)
   c) Deliverable 3: 40pts (+up to 15 extra pts)

   Therefore, the overall possible team grade is 100pts (+up to 30 extra pts). In addition, every team will be considered for one of the three awards: **1) Best Features 2) Most User-friendly 3) Best Design**. The winner team of each award will be awarded another 10 extra pts.

   For the individual grades, we are taking into considerations of individual performances. While we understand that every person has different strengths, you are expected to pull your own weight and not counting on your teammates to carry you through. At the end of the semester, you will be asked to submit a peer review form that evaluates each individual's performance in your team by percentage. These reviews will remain confidential with me. The average of your percentage will be used to calculate your individual grades.

   Out of the 130 possible standard team points, I take 30 points out as the Individual Participation grade. For instance, in a 4-person team, each member's individual grade would be:

   TeamGrade * (100/130) + 30 * (AvgIndPercentage / 25%)

7. **Another word on teamwork:** Teamwork is a crucial skill for a software engineer, and I expect everyone taking this course to take this with utmost seriousness and a collaborative attitude. This project is by no means an easy one, therefore if one member does not apply oneself, the whole team might be negatively impacted.

   Still, I recognize that challenges and situations do happen. Here are my suggestions and policies regarding teamwork issues:
   a) If you think you are falling behind too much, be proactive and seek help from your teammates, classmates, and me. If you still feel overwhelmed, please let your teammates and me know about it, and your decisions about the circumstance. I have had students in the past who simply stopped showing up, and refused to respond to any forms of communication attempts. It certainly did not sit well with the remaining teammates, even if they were able to pull through.
   b) If you are on the other side of the problem, finding yourself (yourselves) always have to carry the team, do most (even all) of the work constantly, also be sure to reach out to me, and we will try to work things out.
   c) While this is an asynchronous online course, since all of you are on Pullman campus, you should set a regular weekly meeting time to keep each other on the same page. Of course, I want to remind you not to do all your coding *only* when you are *physically together*. That is why we are using git – and learning the importance of appropriate good design and task division.

   I hope we don't have to use any of these policies throughout the semester. Nevertheless, they are here to guarantee that if you work hard on the project, you should not have to worry about being treated unfairly.

8. Project resources:
   a) Touhou 10 (the one we are basing our project on):
   http://www.theisozone.com/downloads/pc/windows-games/touhou-project-10-mountain-of-faith-english-by-ferancisco/
   Windows only. You might need certain DirectX 8.0 dlls to run the game. "th10e.exe" is the English patch of the game.
   I strongly recommend trying the easy mode to get a better understanding of the project.
   b) Replays and other assisting files of Touhou 10: see 487-project-sample-resources.zip
   c) Be aware of copyrights. If you use other people's codes, images, music/sound, font, etc., be sure to credit them properly.

3. Project Deliverable 1: see Course Schedule for Due Date. (20pts)

Requirements: You need to build a playable game that is able to reproduce the gameplay shown in this video: 487-Project-SampleGamePlay.mp4

Here are some definitions that are used in this document, with reference to the video's timestamp.

- Player/Player Character
  The character/ that the player can control in the game. Appeared at the beginning at the bottom of the screen. The hitbox, which appears as a white dot in the center of the sprite when "slow speed" mode is activated (see below in required features), is much smaller than the size of the whole sprite.
- Regular Enemies
  Non-boss enemies. There are different variations of such enemies. They move and/or attack by firing bullets.
- Mid/Final Boss
  The Mid Boss appeared at 00:48 mark, and the Final Boss appeared at 01:30 mark. The bosses have more specific movements, and their attacks are more elaborated and complicated.
- Regular Stage
  All gameplay outside of the mid and final boss fight.
- Mid Boss/Final Boss Fight
  Mid boss fight: from 00:48 to 01:15. Final boss fight: from 01:32 to the end of the video. The final boss fight has 4 stages, each of which has a different movement patterns and bullet-firing patterns.

I. The key features **required** are:

1. The player. **(2pts)**
   a) A player character should be able to respond to keyboard control and move in the corresponding 8 directions. Its movements are confined in a certain area on screen.
   b) A player also can switch between "normal speed" and "slow speed" modes. The latter allows the player to move slower and more accurately for dodging.

2. The regular enemies (except bosses) should enter, move and exit in the same (or very similar) quantity, order, movement, and trajectory as shown in the video. **(3pts)**
   a) For the small regular enemies that appeared between 01:11 and 01:14, their appearances are randomized, and your game should duplicate the randomization.

3. The projectiles fired by the regular enemies should also spawn, move and exit in the same (or very similar) quantity, order, movement, and trajectory as shown in the video. In the video, all projectiles fired by the regular enemies spawned at the same position as the enemy. **(5pts)**
   a) They also have identical movement patterns: groups of bullets of one or more (always in odd number) spawned at the location of the enemy, with the center bullet moving directly towards the player's then location at the moment of spawning (sometimes referred to as sniping bullets).

4. The bosses have their own movement patterns and attacks. Your game is expected to imitate them as close as possible. **(10pts)**
   a) The mid boss moves among three spots repeatedly while firing projectiles in a fixed pattern (amounts, angles, speed, etc. are all determined).
   b) The final boss' movements and attacks, on the other hand, has a certain degree of randomization, which means you are not expected, and won't be able to duplicate the identical behaviors. However,

you should imitate these random behaviors to make it as similar as you can.

c) Specifically, the projectiles (bullets, lasers, etc.) fired by the final boss is a combination of random spawning, and patterned movements. Unlike the projectiles by regular enemies, <u>it is helpful to think of the "spawning points" of the projectiles as separate entities from the boss itself</u>.

   For instance, during its first attack (01:37 – 02:22), the red bullets are spawned by two out-spiraling "spawning points" which fires bullets for about one second. All the red bullets are also sniping bullets, i.e., they move directly towards the player's position at the time of spawning. Because of the close spawning, they seem like a winding rope until they move further. Meanwhile, the blue bullets spread in a circle first, and then starts to move in an out-spiraling trajectory.

   The third attack (03:08 – 03:52) is an enhanced version of the first attack: two rings of blue bullets in opposite directions, and the red bullets are spawned 3-way (approximately 120 degrees from each other) with one spawning point for about 2 seconds.

   You should try to identify the similar patterns for the other two attacks. Mathematical skills will be needed for the trajectory calculation.

d) In this video, the bosses go into the next stage or exit after a certain amount of time has passed. Please duplicate this feature as well.

II. <u>The key features **not required** for **Deliverable 1** but you should **start planning ahead** are:</u>

1. The video is a "no shoot" play-through, meaning the player did not fire a single bullet. This is intentional to show the entirety of the stage. In a regular play-through, the player can fire projectiles as well to kill the enemies (hence reduce the number of enemies/bullets and shorten the time needed to survive).

2. The player was hit a few times in the video (02:56, for instance). This happens when the player's hitbox collides with one or more bullet's hitbox.

3. Enemies and Bosses are killed when their HPs (health points) are reduced to 0, causing by damage from the player's projectiles. Bosses may have a separate HP for each stage of attacks. The player on the other hand, functions on a life system as described in the requirements of **Deliverable 2**.

4. Note that in Touhou, projectiles fired by regular enemies remain on the screen when their launchers are killed or exits the screen – unless a Boss Fight starts. On the other hand, the Bosses' projectiles disappeared immediately after each stage ended (either by timeout or losing HP to 0). You are not required to duplicate this behavior for now. But to make your project similar to Touhou, you are advised to implement this in **Deliverable 2**.

The features above required defining hitbox for projectiles, enemies/bosses, and players. For this project, you should duplicate how Touhou hitboxes work, details of which are included below in the requirements for **Deliverable 2.**

III. <u>The features **not required** for **Deliverable 1** are:</u>

1. None of the aesthetic elements is required: music, sound effects, sprites, background, stories/dialogues/characters, etc. For deliverable 1, you may simply use different colored shapes to

represent the entities in the game. A big blue circle as the boss, for instance, is good enough. Of course, don't turn in a blank/black screen as your submission!

2. Scores, power-ups (dropped by the enemies/bosses or when the player was killed), bombs, and other supportive systems.

3. Specific trivial details that are unreasonable for you to follow, such as the exact angle/speed/amount/size-of-hitbox/shape of projectiles, length of time between enemies' entries, movement speed of the player in both normal and slow speed mode, and so on. While the goal is to imitate the gameplay of Touhou, you are not required to actually measure the pixels on a grainy video.

4. However, **10pts extra** are available for the teams who are able to come very, very close to the original, which will be judged based on my game play experience only on the original game and yours. In another word, arts and music/sound do not count here yet.

4. Project Deliverable 2: see Course Schedule for Due Date. (40pts)

Requirements: The main objective of Deliverable 2 is to refactor, redesign and re-implement what you have done in Deliverable 1, with some new required features. You are not required to start all over and rewrite every single line of code of course, but the amount of work will depend on the design quality of your first deliverable. In addition to duplicate the game play in our sample game play video, the new requirements for Deliverable 2 are listed below.

I. The key features **required** are:

1. The hitboxes shall be in place and collision detection shall be functional to allow regular game play. **(10pts)**
   a) The player should be able to fire projectiles and damage/kill enemies and bosses using the keyboard, and the player should be able to be killed by the enemies' and bosses' projectiles.
   b) For the player's projectiles, your implementation can be as simple or complicated as you want. I suggest keeping it simple, and not interfere with the enemies'/bosses' projectiles, i.e. use different textures to avoid confusion, etc.
   c) For enemy projectiles, hitboxes vary depends on the sizes and shapes of the bullets. You are free to define your own in boxes, circles, ellipses, and so on with varying sizes. Note that a laser's hitbox is stretched out through its entire length (see 04:06). Typically, the hitboxes of the projectiles are smaller or the same as the sprite.
   d) For enemies and bosses, the hitboxes are much larger and mostly cover the entire sprite. Killing of enemies and bosses are based on HP calculation: the player's projectiles caused damage by reducing the HPs until it reaches 0, at which point they exit and/or disappear.
   e) The player's hitbox is much smaller than the sprite. The player is hit once its hitbox collides with any other non-player projectiles' hitboxes.

2. The life systems for the player shall be in place. **(5pts)**
   a) The player has an initial number of lives that shall be displayed properly.
   b) Every time the player is hit by a projectile, it loses one life immediately.
   c) If there are still lives left (shown as stars on the right side panel behind "Player"), the player respawns from the center bottom screen. It remains invincible for a few seconds after respawning.

Being hit also cleared the screen of all bullets for a few seconds as well.

d) If there are no more lives left, the player loses the game and the game should show proper prompt for the player to end or exit the game.

e) If the player survives till the Final Boss is beaten (or exits), the player wins the game. The game should also show proper prompt for the player to end or exit the game.

II. Other **requirements** are:

These are not game "features", but are requirements for Deliverable 2. More guidelines and tips would be announced in time for this deliverable.

1. You are required to use appropriate design patterns to implement various aspects of the game. Basically you should utilize the design patterns you learnt from the course at this point. **(10pts)**

2. Basic concepts of an overall architecture of your game should also be in a reasonable, if not finalized or mature shape. **(5pts)**

3. A design document that includes the following contents: 1) design quality attributes applicable to the game and your solution; 2) UML class diagrams depicting design patterns you used for the game, and explain the reason for choosing said design patterns; 3) UML component diagrams that shows the architecture of your game – for simplicity purposes, it should be either a multi-layered architecture, or a Model-View-Controller architecture. Proper components shall be defined in each of the subsystems in your architecture. **(10pts)**

III. The key features **not required** for **Deliverable 2** but you should **start planning ahead** are:

1. Power-ups (dropped by the enemies/bosses or when the player is hit), bombs, and other supportive systems are trademarks of such games. You should plan ahead to implement at least one of the supportive systems, which will be required in the final deliverable.

2. An easily activated cheating mode. (Why? Discuss.)

3. Making abstractions from all aspects of the game you have implemented: enemies, bosses, bullets; movement patterns; etc. In Deliverable 3, you will be tasked with designing the level with all possible abstractions you can think of. Again, the challenge of this task will depend heavily on your design.

IV. The features **not required** for **Deliverable 2** are:

1. Aesthetics are still not required at this point, as long as the game is in general playable.

2. The specific values of HPs (enemies and bosses) and damages (player projectiles) are up to your team. Same goes for the actual size/shape of hitboxes of enemies/bosses/bullets. They should make sense in a normal game play though: for instance, making the boss's HP 1 and the player's damage 1000 would not be reasonable. (*Unless you played Undertale.)

3. Difficulty level. The sample video played in the "Easy" mode of Touhou, which has a total of four difficulty levels: Easy, Normal, Hard and Lunatic. I included several replay files in 487-project-sample-resources.zip. You may view them from the "Replay" option of the menu in Touhou 10 if you've

downloaded it.

4. Another **5pts extra** are available for the teams who construct a functional menu within the game, and allows for basic key configurations. For instance, allowing the player to use WASG instead of arrow keys. It’s a simple function, but make sure your solution is “elegant” in your design.

5. Project Deliverable 3: see Course Schedule for Due Date. (40pts)

Requirements: The main objective of Deliverable 3 is to build a level interpreter for the game, that is able to reproduce all AI behaviors in your game, specifically, the type/movement pattern/attack pattern of all enemies, bosses, and their projectiles. In addition, your final product should be completed in all designs and features.

I. The key features **required** are:

1. An in-game engine (or even a stand-alone program) that functions as a "level interpreter" of the game. **(30pts)**

a) The engine shall be able to read in an external file (i.e. in XML/JSON or other formats as you see fit – for examples, see http://json.org/example.html), and then interprets the file to dictate the AI behaviors throughout one level of the game.

For instance, you might define a script named "stage1.json", in which you wrote:

```
{
   "waves":
   {  "wave":
      {
          "id": "1",
         "time": {"0", "1000"},
         "enemyType": "A",
         "enemyAmount": "5",
         "interval": "200",
         "enemyBulletType":
         {
            "color": "red",
            "type": "B",
            "amount": "20",
            "speed": "10"
         }
      },
      "wave":
      {
         "id": "2",
         "bossType": "Boss1",
          ......
       }
   }
}
```

You should be able to load the script from the game and start a level. The "AI" should behave accordingly to your script by deploying, for instance, "waves" of enemies that moves a certain way, looks a certain way and fires certain types of projectiles a certain way.

This is just an example, and the "wave" definition is mere an inaccurate suggestion on how to represent the level. Make sure to define and design how you want to model your level accurately and effectively. Nevertheless, here is the minimum list of info that should be defined and customizable in your script:

i. The type of enemies to spawn; when/where/how/how many do they spawn, etc.
ii. The type of movements of the enemies/bosses.
iii. The type of bullets the enemies/bosses "fire"; when/where/how/how many do the bullets spawn, etc.
iv. The type of movements of the bullets.

The "type of movements" is highlighted as a hint on how you should think about the abstractions at this stage. Simply put, the goal is to make as many AI-related elements in your game customizable as you can. This means there should be as little "hard-coded" information in your source code as possible.

b) Write a script that would allow your engine to start a level that duplicates the game play shown in the sample game play video: 487-Project-SampleGamePlay.mp4

2. Implement at least one of the following supportive systems (not including scoring system). You are free to define your own rules regarding these systems and you don't have to follow the Touhou rules. (It is a good place to start for inspiration though.) **(10pts)**
   a) Power-ups: dropped by enemies/bosses when they are killed, or when the player dies. The player can pick up the drops and increase the damage and/or amounts of its projectiles.
   b) Bomb system: when the player is in danger, the player can "throw" a bomb to get out of trouble. For instance, the bomb can clear all bullets in a certain area (or the whole screen), deals massive damage to all enemies and/or bosses, grants the player invincibility for a short period of time, etc.
   c) Reward system: the player can sometimes pick up life/bomb pieces to increase its life/bomb counts.

II. The features **not required** but can earn you **extra points** (up to 15pts) are:

1. Aesthetic make-overs: full on sprites, backgrounds, animations, sounds/music, etc. **(5pts)**

2. Easily activated cheating mode. **(5pts)**

3. Difficulty level design: for your game, engine and script, find a way to add additional difficulty levels that the player can play. **(5pts)**
   a) If your team is able to duplicate the Lunatic mode game play of Touhou10's stage 1, **10pts**

4. Use your level interpreter engine and write another script to design a second level after the original one. **(5pts)** You should aim to include more types of bullets movement patterns for instance.
   a) If your team is able to duplicate Touhou10's stage 2 game play on easy mode, **10pts**

5. Other additional features you come up with that you think are worthy of rewards.