Game Proposal: Red vs Blue Arena

CPSC 427 – Video Game Programming

Team: Team 22

Jiakang Huang 66382623

Malcolm Zhao 11913985

Edward Wang 88723788

Junkai Ding 74511775

Kelly Wong 20721783

Story:

Briefly describe the overall game structure with a possible background story or motivation. Focus on the gameplay elements of the game over the background story.

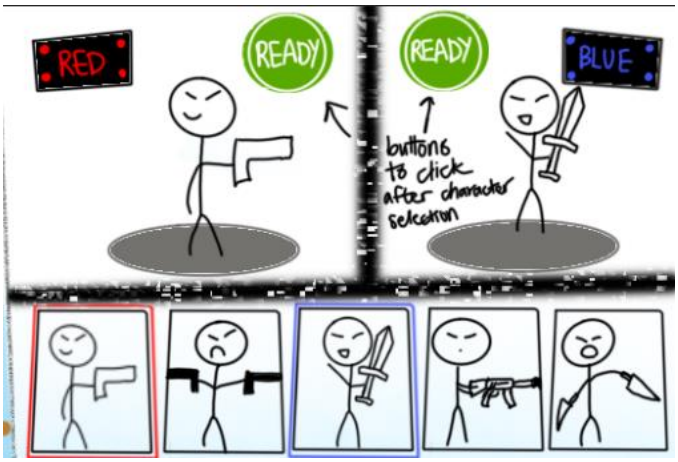
In a fantasy world, the red and blue nations are locked in a century's old rivalry. Each nation selects a champion to fight in one-on-one duels in a deadly arena. Two players will control a champion from each nation as they battle each other in the arena, for glory. To spice up the battlefield, the arena will include a variety of features such as items, a deadly laser, and portals to keep gameplay dynamic and fresh. The arena is certainly a dangerous front where steel and guns collide, and one nation will collapse.

Scenes:

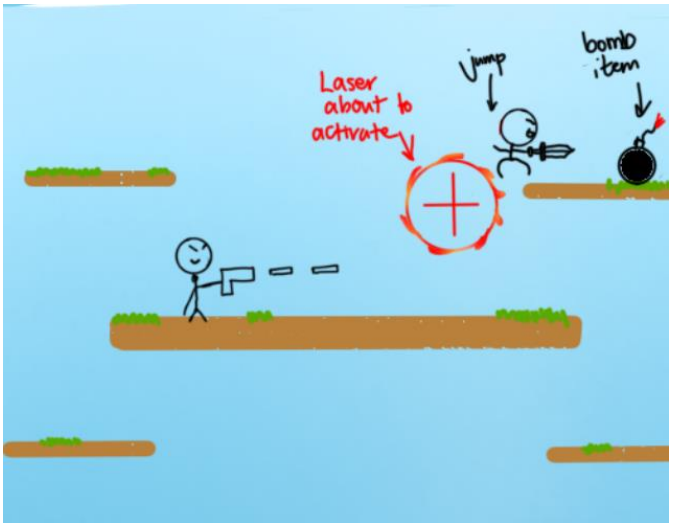
Produce basic, yet descriptive, sketches of the major game states (screens or scenes). These should be consistent with the game design elements, and help you assess the amount of work to be done. These should clearly show how players will interact with the game and what the outcomes of their interactions will be. For example, jumping onto platforms, shooting projectiles, enemy pathfinding or 'seeing' the player. This section is meant to demonstrate how the game will play and feeds into the technical and advanced technical element sections below. If taking inspiration from other games, you can include annotated screenshots that capture the game play elements you are planning to copy.

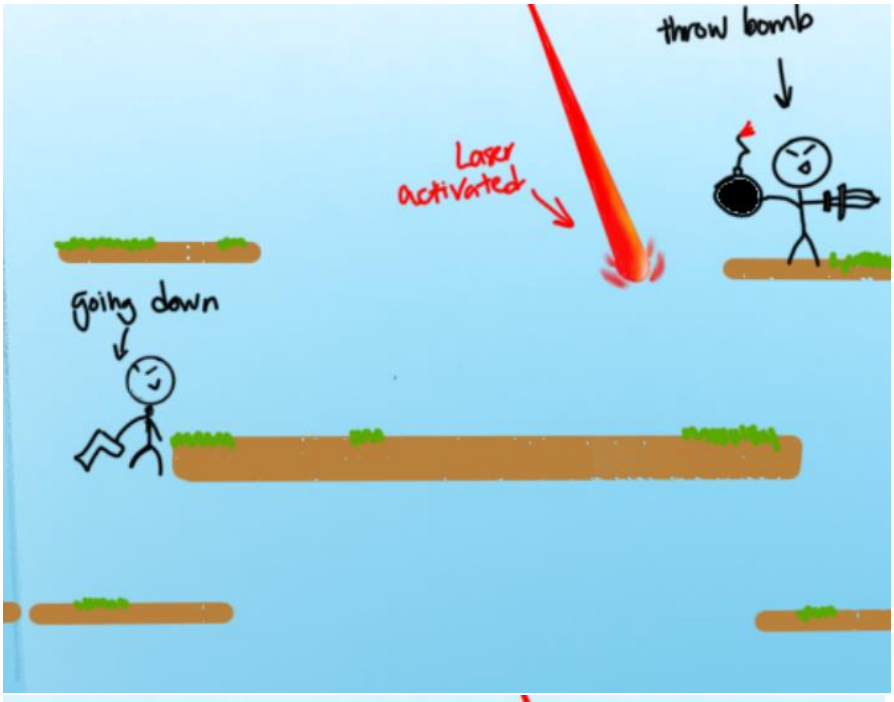
Home Page/ UI

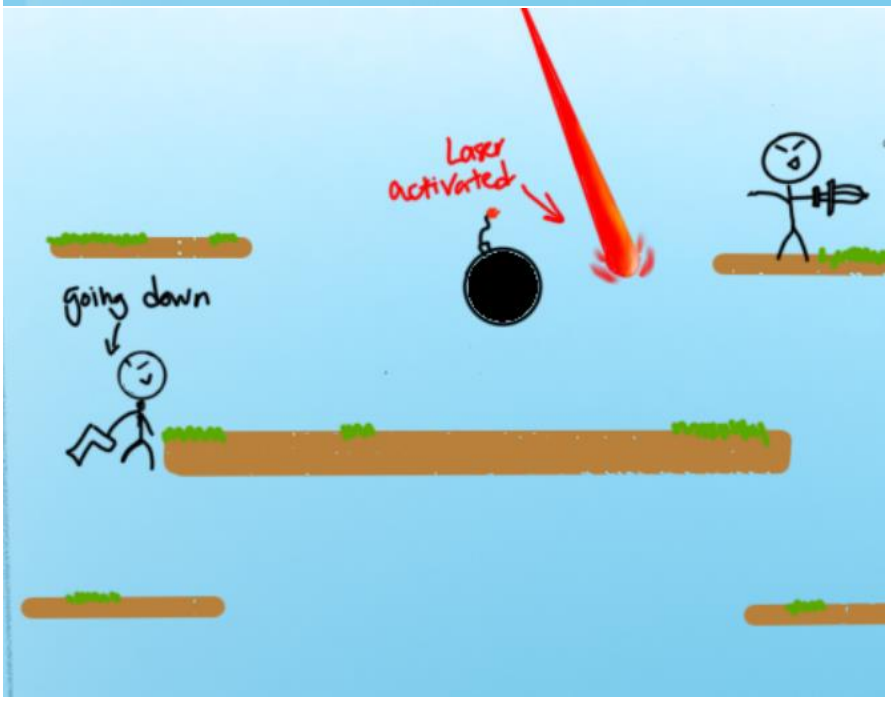
Character Selection / Team Selection

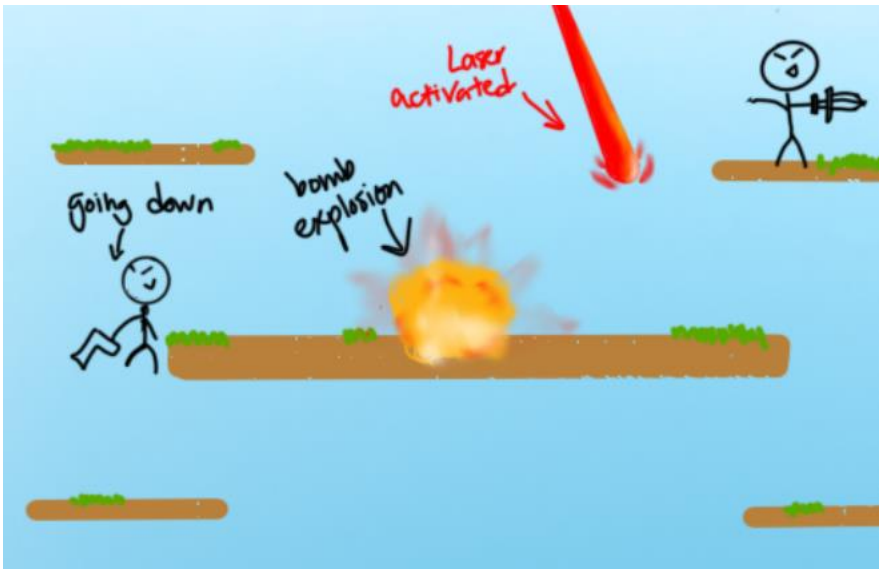


Gameplay




throw bomb
Laser activated
going down


Laser activated
going down



Technical Elements:

Identify how the game satisfies the core technical requirements: rendering; geometric/sprite/other assets; 2D geometry manipulation (transformation, collisions, etc.); gameplay logic/AI, physics.

- Rendering: The game is designed as a 2D game, with an emphasis on maintaining at least 60 FPS for smooth gameplay. Shaders can generate particle effects for actions like gunfire, explosions, or when swords clash. These effects add excitement to the game and make interactions more visually engaging.
- Geometry/Sprites: We will assign animations to characters in different states, such as when they are idle, attacking, and jumping. Projectiles can also have various animations such as how they disintegrate after hitting a wall, or how they disappear after striking a player.
- Gameplay Logic: Bullets that collide with each other will cancel out. When a player is hit, their health will decrease, and the game ends when one player's health reaches zero. Players should interact with stage blocks by being able to stand on them, move on them, etc. without falling through. How players are controlled is explained in depth in the Devices section.
- AI: An AI-controlled laser will follow the two players and shoot at them when directly overhead. When a player touches the laser, their health will decrease.
- Physics: The primary physics mechanism involves gravity, which pulls characters down when they fall. Items such as grenades should also be affected by gravity. Other physics related mechanics include the portal which teleports the player or projectiles position to another part of the arena.
- Sound Effects: The game will feature sound effects for actions such as gun shooting, sword swinging, and background music, which can be either sourced online or created.

Advanced Technical Elements:

List the more advanced and additional technical elements you intend to include in the game prioritized on likelihood of inclusion. Describe the impact on the gameplay in the event of skipping each of the features and propose an alternative.

Complex Character Animation:

- Description: include more complex animation (for example, more frames for jumping, attack, move, and when character falls onto ground, there will be dusting effect.)
- Likelihood of Inclusion: Medium-High
- Effects: characters' movement would look smoother on screen If skipped, should not impact fundamental gameplay beyond characters movement may look more static & skating on screen.

Advanced AI for traps/power-ups:

- Description: Implement more complex AI for traps and power-ups such as dynamic movement, pattern recognition, and adaptive difficulty (e.g., traps move faster through the game)
- Likelihood of Inclusion: High
- Effects: This would make traps and power-ups more unpredictable, adding another layer of Strategy.

Dynamic Weather and map:

- Description: Include dynamic weather or maps which have different effects on gamers during the fight, which may affect visibility, movement, or abilities.
- Likelihood of Inclusion: Low-Medium
- Effects: It adds randomness to each game, increasing playability.
- If Skipped: The visual atmosphere might feel static, but gameplay wouldn't be impacted beyond lack of environmental diversity.
- Alternative: Simple, static background differs through different maps.

Weapon Customized/upgrades:

- Description: Enable user to customize weapons before the game and upgrade them during the Game by power-ups.
- Likelihood of Inclusion: Medium
- Effects: It adds strategic depth by giving players more control over their playstyle and increases replay ability.
- If Skipped: Weapons would remain fixed, but still allowing the game to focus on core mechanics.

- Alternative: Offer predefined set of weapons and powers, players can still select before match.

Interactive Environment Objects:

- Description: Add objects that the player can interact with, such as barriers that can block bullets, Power-ups to upgrade weapons.
- Likelihood of Inclusion: Medium
- Effects: It adds strategic depth by allowing players to manipulate the environment for cover or item discovery.
- If Skipped: Environment will stay static, with fewer interactive elements, but core gameplay will remain unchanged.
- Alternative: Use environment hazards without player interaction.

Advanced Sound Design/Directional Audio:

- Description: Implement directional audio, where the sound of weapons, traps, or power-ups shifts based on player position relative to the source.
- Likelihood of Inclusion: Medium-Low
- Effects: It enhances the immersive experience, giving players audio clues about nearby threats or opportunities.
- If Skipped: Players rely more on visual cues, which could make gameplay slightly less immersive but not game breaking.
- Alternative: Include simple, non-directional sound effects that trigger based on player actions.

Devices:

Explain which input devices you plan on supporting and how they map to in-game controls.

The idea is for the game to be played on a single PC/laptop, with a keyboard. For the players, player one will use the WASD keys to move, crouch, and jump, while using the keys above which are the “1”, “2”, and “3” keys to attack, switch weapons, and use items, respectively. Similarly, player two will occupy the right side of the keyboard, using the arrow keys to move, crouch and jump, while using the keys above being “,”, “.”, “/” to attack, switch weapons, and use items.

Tools:

Specify and motivate the libraries and tools that you plan on using except for C/C++ and OpenGL.

As for now, we have considered the following tools to be potentially useful in completing the project:

- Rendering: OpenGL

- Sound: OpenAL, FMOD or irrKlang (for better effects)
- Physics: basic: Box2D, for consideration: Bullet Physics
- Shader Development: GLSL Validator for checking GLSL errors
- Game AI: OpenSteer or Recast & Detour or AIPatrol
- 2D geometry: GLM or Eigen

We will likely not use all of these, but we are more inclined to use libraries to assist us in achieving tasks that are hard to complete from scratch i.e. sound design and sound effects, to which we are highly considering using the irrKlang library.

Team management:

Identify how you will assign and track tasks and describe the internal deadlines and policies you will use to meet the goals of each milestone.

We consider that a good working ethics is to start early and avoid late-minute communication confusion such that we want our work done at least 1 day ahead of the deadline; in addition to that, we also try to split up tasks equally and make it fair for everyone to contribute. Thus, following the team contract, our team will meet up every Tuesday and Friday plus the Tutorial time allotted to us on Wednesday if applies. If during any of those meetings, we identified any difficulties in meeting the deadlines, we will use Sunday from 1-2PM as a backup time for emergencies.

As for the scope and ensuring that we finish the game on time, we currently have ambitious plans such as having a variety of stages to showcase different features we plan to implement, as well as designing 4-5 different melee or ranged weapons for the players to utilize. However, at the core we'll focus on designing one stage to be good first, and design other stages as time permits throughout the term. Same with weapons, ensuring we have two basic weapons, being a working gun and sword, is our utmost priority to finish before we branch out and make more weapons.

General game development such as programming the player and projectile entities, stages, and obstacles will likely be a lot of work, to which we have assigned two people to be general game devs. We will assign one person to handle the interface side of things, as in the UI and graphics/animation. More niche requirements like physics, collision handling, etc. will be divided based on the preference of the remaining group members. Overall, the roles are assigned as follows:

Name	Roles
Edward Wang	Game Dev.

Malcolm Zhao	Game Dev.
Kelly Wong	UI and Graphics/Animation
Junkai Ding	Physics & AI
Jiakang Huang	Collision Handling, Sounds; assist in UI and graphics based on circumstances

Development Plan:

Provide a list of tasks that your team will work on for each of the weekly deadlines. Account for some testing time and potential delays, as well as describing alternative options (plan B). Include all the major features you plan on implementing (no code).

Milestone 1: Skeletal Game

Week 1: In the first week of creating our game, we'll program the most vital entities of our game, such as the players, projectiles, blocks that make up a stage, etc. As well, we'll work on basic rendering to display our entities.

Week 2: In week two, we'll work on linking keyboard inputs to reflect movement in-game, as well as vital gameplay logic in the form of collision handling, such as when a player collides or a bullet or two bullets collide with each other.

Below is a table for how we plan to divide tasks, as well as when we want them to be finished by, for milestone 1.

People	Task	Date
Edward	Basic Assets	Oct. 1st
Kelly	Rendering	Oct. 3rd
Junkai	Rendering	Oct. 3rd
Malcolm	Input Reponses	Oct. 3rd
Jiakang	Collision / event-driven/random responses	Oct. 4th

Milestone 2: Minimal Playability

Week 1: In case we have game breaking bugs from M1, we'll assign one game dev to fix those bugs. Also, we'll assign our game AI person to start working on implementing the laser and item systems - this should take both weeks for them to do. The team member designated to work on collision handling will start refactoring M1 the code to be mesh

based. The remaining game developer and UI person will work together to make a more comprehensive UI.

Week 2: After the UI is improved, we will further make the game nicer by working on our assets and animation. These two areas may take a while, so we'll assign all available members of the team to work on this. Before the end of the milestone, we'll assign one game developer to record all known bugs, and the other to make a user tutorial.

Milestone 3: Playability

Week 1: In the case that things like collision handling and sound design is done, we'll reallocate the member assigned to those tasks to work on UI and graphics instead. From here on, we anticipate the core UI to be done, and we'll allocate the UI and graphics people to work on rendering new features we implement as they come out. One game developer will be the lead in designing new features, while another will focus on fixing bugs and preventing memory leaks, while also designing new features as time permits.

Week 2: In the second week, we'll focus on satisfying the implementing advanced technical requirements to fulfill the creative component. UI people will implement physics based animation and complex prescribed motion as necessary. Other team members are also interested in AI, and we are looking to focus on satisfying the creative complex via flashy AI methods if possible.

Milestone 4: Final Game

Week 1: At this point, we'll focus on implementing new features as time permits. This will come in the form of releasing new stages, designing new weapons, adding more items in the item system, etc. To add freshness to gameplay. UI people work on rendering new features as they come out.

Week 2: In the final stretch, we'll polish the game to satisfy the creative component, likely through means of graphics/UI/UX improvements, i.e. drawing cutscenes as a UX feature.