CS 488 Project Proposal

Title: The King of Cute Fighters

Name: Xiling Wu

Student ID: 20561976

User ID: x242wu

## Final Project:

#### Purpose:

The application will provide player(s) a 3D version of “The King of Fighters” with characters as carton roles. The player will be able to choose from build-in cartoon characters and fight with other characters, practice combo skills and response time.

The basic game logic is the same as the “The King of Fighters”. The player could move the character with arrow keys and use four other keys to attack.

#### Statement:

The game is about control a cute cartoon character to defeat other ones.

Player will first pick a character and an opponent, then two characters will be placed into a room (with random background). Player then use arrow keys and four other keys to move and attack. The one first loses all life value will lose the fight.

This project will be implemented based on A3. An MVC framework needs to be introduced for easy game logic design and user input handling. All the Node classes in A3 also need to be updated in order to achieve complex movement animation and collision detection. A few physic models are also necessary to handle gravity and inertia. Character model needs to be detailed, so new material and texture are also required.

Potential challenge:

Arm and leg hit movement design is challenging since it requires a smooth animation which means the joints need to move synergistically. And the implementation of the skill combo is also a challenge since the program needs to analyze the input buffer and give character movement instruction correspondingly. All these needs to be done smoothly without any obvious delay also requires a well-designed process flow.

Most of the existing fighting games are in 2D, and I think being able to control a carton character fighting in a 3D scene would be very interesting.

Things to learn:

1. Build a physic engine involves gravity and inertia in the graphics system.
2. Implement collision detection systems.
3. Good design approach to build a character control system
4. How to build smooth animation

#### Technical Outline:

##### Modelling the Scene:

The characters will be placed in a 3D scene that is surrounded by objects like walls that are used to maintain the boundary of the scene, and character can move freely within the scene. An appropriate background will also be included.

##### UI:

The game UI will contain useful function like restart, pause, exit, etc. Two life value bars will also be displayed to show the remaining life value.

##### Animation:

The character will be able to hit with arm/leg or jump smoothly. So an appropriate animation system needs to be implemented. Loading and victory/defeated scene may also require animation. I plan to build an animation class which will be able to take an instruction such as “hit with arm” as an input and call the update function on the corresponding node with an appropriate time gap.

##### Collision detection:

One key feature is to detect if one character’s attack hit another one or not. So a dynamic collision detection system is necessary. I found a useful resource about using sweep test for easy collision detection mentioned by Kasper Fauerby in his paper [1].

##### Sound:

To build a good audiovisual system, three key components need to be considered: space/environment, different objects, and symbols [2]. In order to achieve those goals, an appropriate background sound will be added for better user experience, and when character hit each other, appropriate sound effect will be played to give user auditory feedback.

##### Physics engine:

Since characters are able to move (including jump) and try to hit each other, the physics engine is necessary to make sure the movement seems natural. Gravity will be introduced in two parts. The first part is the jump movement. The speed of character should decrease while moving up and increase while moving down. The second part is when one of the characters is defeated, it should fall down because of gravity. As for inertia, after user stop pressing arrow key, the character should gradually stop its movement.

##### Texture mapping:

For the character, each node will be mapped with color and texture other than default ones to make the character more vivid. And for the floor and background, the customized texture will also be mapped.

##### Particle system:

When a character successfully hit another character, a visible particle effect will be displayed in order to give user visual feedback. The particle effect will be displayed at the node being hit and last for a few seconds.

##### Motion blur:

When character moves fast or use certain skills, a motion blur effect should be triggered for better visual feedback.

##### Shadow:

Shadow will be displayed under each character’s feet. The shadow will be implemented using shadow mapping with the depth map.

#### Bibliography:

[1]. Kasper Fauerby. Improved Collision detection and Response. <http://www.peroxide.dk/> pp. 14 - 17

[2]. Jarvinen, A. (2002) ‘Gran Stylissimo: The Audiovisual Elements and Styles in Computer and Videogames’ In: Mayra, F. (ed.) Proceedings of the Computer Games and Digital Cultures Conference. Tampere, June 2002.

[3]. Kamat, V. R., & Martinez, J. C. (2002). Scene Graph and Frame Update Algorithms for Smooth and Scalable 3D Visualization of Simulated Construction Operations. Computer-Aided Civil and Infrastructure Engineering, 17(4), pp. 228-245.

[4]. Rekapalli, P. V., Martínez, J. C., & Kamat, V. R. (2009). Algorithm for Accurate Three-Dimensional Scene Graph Updates in High-Speed Animations of Previously Simulated Construction Operations. Computer-Aided Civil and Infrastructure Engineering, 24(3), pp. 186-198.

#### Objectives:

Full UserID: x242wu

Student ID: 20561976

\_ \_ \_ 1: **Modelling the scene:** A scene that represents the fighting room should be rendered.

\_ \_ \_ 2: **UI**: User should be able to start, pause, restart the game and able to see the life value bar for each character.

\_ \_ \_ 3: **Animation**: Character movement should be smooth and natural.

\_ \_ \_ 4: **Collision detection**: Character should be able to hit the other one with the attack.

\_ \_ \_ 5: **Sound**: background sound added and hit sound correctly synchronized with movement.

\_ \_ \_ 6: **Physics engine**: Gravity and inertia systems should be implemented.

\_ \_ \_ 7: **Texture mapping**: Characters and background should have texture being displayed.

\_ \_ \_ 8: **Particle system**: Particle effect should be visible when character is hit successfully.

\_ \_ \_ 9: **Motion Blur**: Motion blur on moving parts should be visible on certain scenarios.

\_ \_ \_ 10: **Shadow**: Shadow should be at the correct location(under character’s feet).