Kung Fu Master

Learning via interactive VR in Unity

Yu Zeng zengyu@stanford.edu
Xiaole Wang xiaolew@stanford.edu

1 Project Outline

This project is a VR game which aims to help people learn (play) Kung Fu. Our game is developed in Unity3D and based on HTC Vive and two controllers.

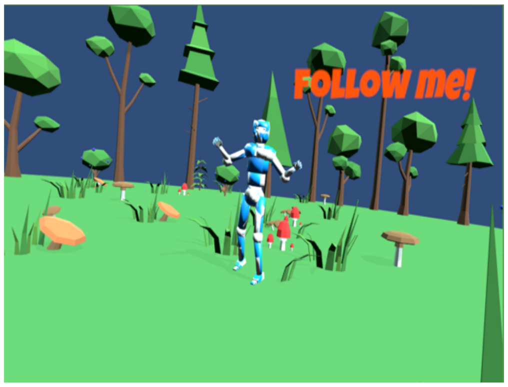
In general, we track the collisions of the controller to see if the player hits any target. If the player hits something, the target will change the color or change the position depends which target is hit. When the player starts the game, the player could warm-up by doing some exercise and learn the movements. In the game stage, the player can earn points by hitting some target in terms of the accuracy and the speed, faster earns more!

2 Game Design

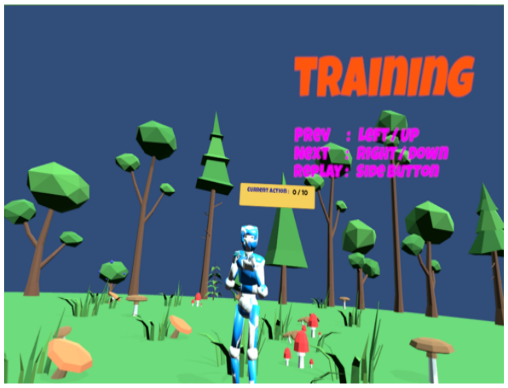
Our game design concept is to help the player learn Kung Fu movements and enjoy the game. So we designed the guided learning system and trajectory based score system to achieve this goal.

2.1 Guided learning system

Our game includes three stages to help the player learn and play. First is the warm-up stage, by following the virtual tutor, this exercise let people do some gentle stretching exercises to prevent stiffness. The second one is training stage to teach the player basic movements one by one, the player can also move to the previous movement or repeat the current one to make sure he/she knows how to do it. The third one is the game part, the player will get scores by hitting the target.



(a) Stage 1: Warm-up



(b) Stage 2: traning

2.2 Score system design

Instead of hitting a single target to get scores, we design 4 different trajectories for each hand, and we have ten target balls along the trajectory with an ending cube. We could denote these eight paths as

$\mathbf{P}_i^{\text{left}}$and $\mathbf{P}_i^{\text{right}}$, where $i \in \{1, 2, 3, 4\}$. For a specific path in either hand, it can be represented as

$$\mathbf{P}_i = \{b_1, b_2, \cdots, b_{10}, c\}$$
$$|b_j| = 1,\ |c| = 5$$

where $b_j, j = 1, \cdots, 10$ means the j-th ball and the final c is the ending cube. Moreover, each ball is worth 1 point while a cube is 5 points.

For example, if the player only hit the k-th ball, we would

1. gray previous $b_1, \cdots, b_{k-1}$ balls out.
2. score the current k-th hit by 1 point.

This hit left $b_{k+1}, \cdots, b_{10}, c$ objects active for scoring.

Similarly, if we continuously hit $b_k, \cdots, b_{10}, c$, the player would get $1 \times (10 - k + 1) + 5 \times 1$ points.

All 10 balls plus with a cube will be reset to a new random trajectory once we hit the last cube. In the limited time (30 seconds), the player makes the movements faster and more accurate, he / she will get more scores.

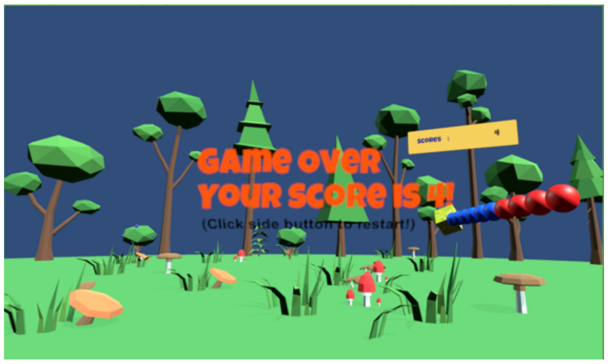


Figure 2: Right-hand straight punch trajectory

3 UI Design

3.1 UI components

In the beginning of every stage of the game, the player will see an instruction floating in the center which told us to pull the trigger to continue. In the training stage, the player can push the button to adjust the step the player wants to learn, and a panel near the tutor will show which step he is teaching. In the game stage, a red count-down text will show the time left, and a score panel will show how many scores the player has earned. When the count down (30 seconds) is over, a message will show the score and ask if the player wants to push the side button down to restart the game.

3.2 Head-tracked position

When the player starts the game, we want to make sure all the instructions and tutor are all in sight with a fixed distance. To do this, we can simply use the forward vector provided by Unity3D and add some distance shift. That is, $P_{\text{tutor}} = P_{\text{camera}} + \text{distance} \cdot \mathbf{u}_{\text{forward}}$. However, if we also want some horizontal offset with the same distance, we then use the quaternion taught in class:

1. Get the forward vector of the camera (headset), denoted as $\mathbf{u} = (u_x, u_y, u_z)$ and the converted quaternion $q_{\mathbf{u}} = 0 + iu_x + ju_y + ku_z$.
2. Construct the rotation quaternion by an axis-angle form where the axis is the up vector $\mathbf{v} = (0, 1, 0)$ and the angle is the Euler angle around y (how much do we want the target to be shifted). We mark corresponding quaternion to be q.
3. At last, rotate $q_{\mathbf{u}}$ by quaternion q is the rotated $q'_{\mathbf{u}} = qq_{\mathbf{u}}q^{-1}$.

4 Assets We used

Melee Boxing Animations. We used this asset for the tutor to show the movement for the player.

Scenes. We used the low poly forests to build the scene considered considering the rendering burden.