Innovative & Immersive Fitness Experience

Environment Integration Manual

Created For

Sense | Think | Act

Company's Contact: Farjana Salahudin | Khalid Al Ali

Advising Professor: Selma Limam Mansar

Course: Information System Consulting Project | Spring 2023

Team Members: Ahmed Issaoui | Jewel Dsouza | Mariam Al Thani | Raghad Abunabaa

Date: Thursday 13th April 2023

Environment Integration Manual

Purpose

This document serves as an encompassing manual on the integration between the environments and the remaining mechanisms of the system that makes up the immersive fitness experience. From this manual, you will be able to understand the relationship between the environment background and the Python-created aspects of the system, how to change the background of the experience through the Python code, and how to alter the environment's presentation.

Description of the relationship between Python & Environment Background

Currently, the system behind the experience is split into two parts. the environments are created on TouchDesigner and the remaining mechanics are created on Python, mainly through a Pygame base. The three designs created on TouchDesigner have the same dimensions as the screen presented on Python (1280x720), and are exported as separate mp4 videos that are stored in the same file directory as the Python file. The videos can be of varying lengths in time and this does not and should not affect experience presentation (unless this is something that one would like to purposely achieve in future development, visit the alteration section).

The videos of the environment, which we will refer to as the environment or the videos interchangeably, are then integrated into the Python code as follows.

Firstly, the video, or the file name of the video, is captured to store information on its size and length.

```
video = cv2.VideoCapture("background2.mp4")
```

Then, a window the same size as the video is created - this goes on top of the screen that is presented and allows both to be the same size. Here, the file name of the video is all that needs to be identified.

```
window = pygame.display.set_mode(video_image.shape[1::-1])
```

Additionally, a numerical variable, ‘frame_counter’, is created and set to equal 0. This is created for the purpose of looping the video that will be playing in the background. This variable increases by one with each event while the camera is opened. So, constantly.

The actual code that plays the video is as below. Firstly, check if the system was able to locate and read the video. In the case that it does, we turn that video into a pygame surface object titled ‘video_surf’ to allow us to present that video alongside the rest of the screen and objects on the screen.

```
  success, video_image = video.read()

  if success:

     video_surf = pygame.image.frombuffer(video_image.tobytes(), video_image.shape[1::-1],
"BGR")
```

After this, we simply present this video (now a pygame surface) on the window, starting from the top left corner, with (0,0) coordinates.

```
window.blit(video_surf, (0, 0))
```

The video is looped through the following code. It first is triggered when it has been identified that the number of frames passed is the number of frames in the video itself. If this is the case, we simply reset the frame counter and re-capture the video which loops it by playing it from the beginning.

```
 if frame_counter == video.get(cv2.CAP_PROP_FRAME_COUNT):

     frame_counter = 0

     video = cv2.VideoCapture("background2.mp4")
```

Additional information and alterations will be made accordingly upon implementation of the three environments and the systems random choice between the three environments. In this case, a string variable with the file name will be stored in the beginning of the code and a randomizer will assign the variable one of the three file names. The rest of the aforementioned code will remain the same.

Alterations to video presentation

Speed

To change the speed of the environment in the background, one simply has to change the framerate, which is set at 120. This is found at the bottom of the code and is as follows. Increasing the framerate will speed the video up and decreasing the framerate will slow the video down.

```
clock.tick(120)
```

Looping

To modify the looping aspect of the video, for example, to loop once we reach the penultimate frame, one simply has to change the if statement that resets and recaptures the frame_counter and video, respectively.

```
if frame_counter == video.get(cv2.CAP_PROP_FRAME_COUNT):

    frame_counter = 0

    video = cv2.VideoCapture("background2.mp4")
```

Randomization

Additionally, to alter the randomness of the environment presentation, one can alter the code presented in the looping section above to reset the file name variable to be a different file name. Thus, once one video ends, it will switch it another (or the same one, depending on the random choice).