

*Hard-working is the key of success.
Never, never, never give up.*

Projects and Researches

Presenter:

Abduraimjonov Abdurakhmon

Agenda

I. Researches and Projects in **Onycom**

1. Loading Time Project
2. Creating Automation Testing Tool for mobile applications
3. Detecting clickable objects in mobile applications
4. Creating Auto-labeling tool for labeling all objects (buttons, text buttons, image and etc..) on Image
5. Multi-devices controlling in the same time

II. Researches and Projects in **Roborus**

1. AI Projects and Researches
2. GUI application Development

III. Researches and Projects in **GREW Creative Lab**

1. CSONG Development

IV. Researches and Projects in **CVPR Lab**

1. Computer Vision Researches
2. AI Researches and Paper summary

Loading Time Project

I. Creating Loading time **detector** and **classifier**.

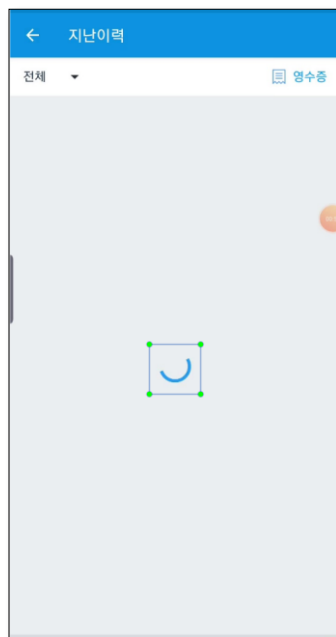
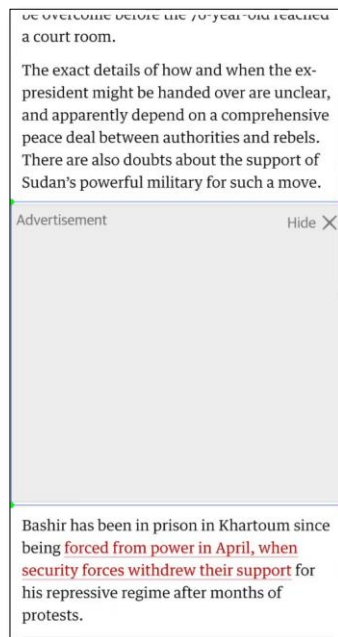
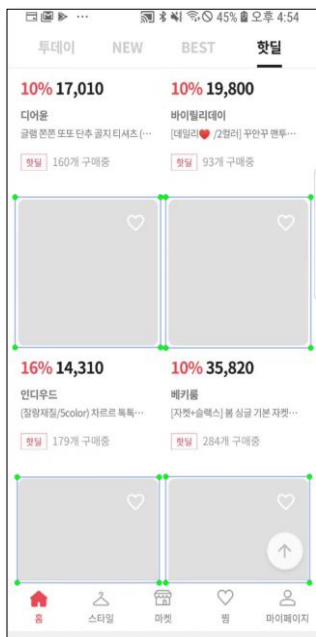
II. Creating Loading time API.

Step 1 > Collecting dataset: Take screenshots using **150** mobile apps

Step 2 > Labeling dataset using **LabelImg** tool (link: <https://github.com/tzutalin/labelImg>)

class names = [loading]

Loading image examples:



Loading Time Project

Creating Loading time detector and classifier.

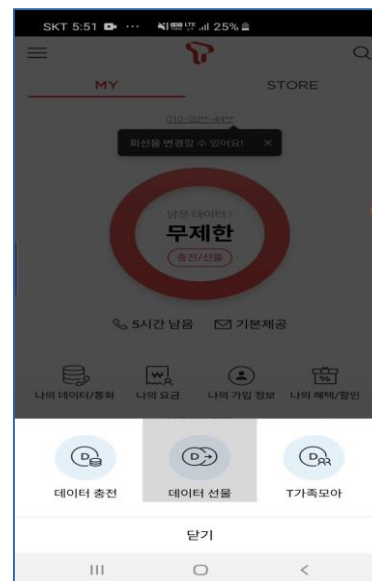
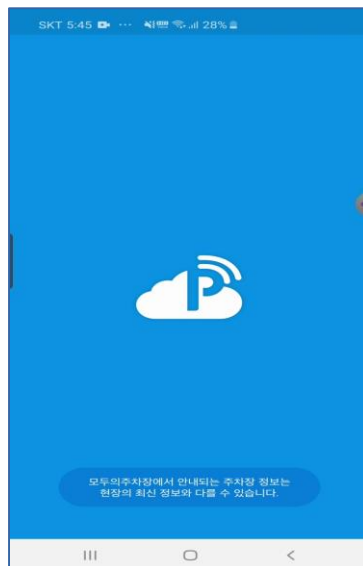
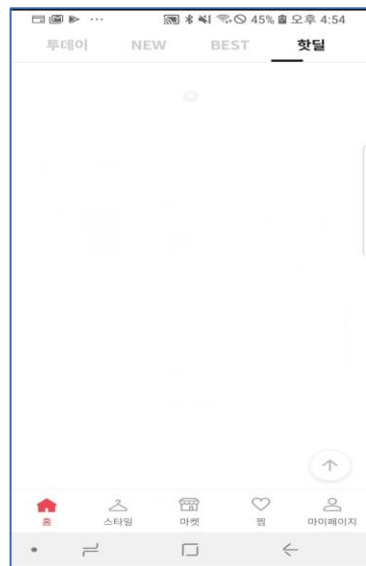
Creating Loading time API.

Step 1 > Collecting dataset: Take screenshots using **150** mobile apps

Step 2 > Labeling dataset using **Labellmg** tool (link: <https://github.com/tzutalin/labellmg>)

Step 3 > Sorting dataset for classification:

class names = [**loading**, **normal**]



Loading Time Project : Loading Time API

Creating Loading time detector and classifier.

Creating Loading time API.

Step 1 > Collecting dataset: Take screenshots using **150** mobile apps

Step 2 > Labeling dataset using **LabelImg** tool (link: <https://github.com/tzutalin/labelImg>)

Step 3 > Sorting dataset for classification

Step 4 > **Making detection** and **classification model**:

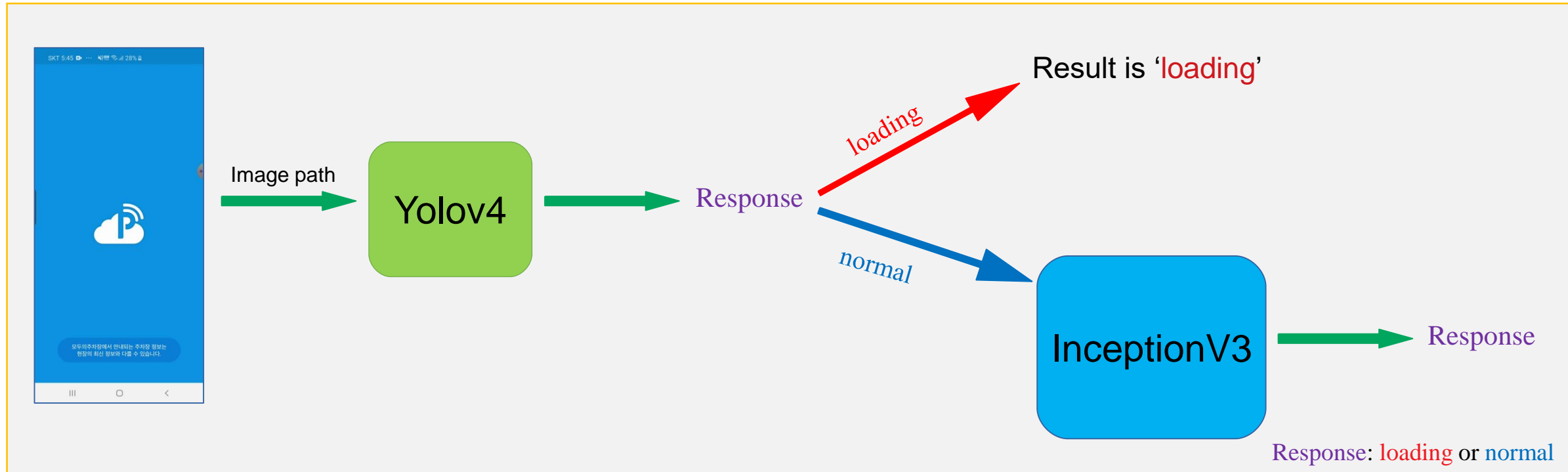
Detection model: Training dataset with **Yolov3** and **Yolov4** using Darknet (link: <https://github.com/AlexeyAB/darknet>)

Classification model: Training dataset with **Inception v3** pre-trained model (link: https://tfhub.dev/google/imagenet/inception_v3/feature_vector/3)

Step 5 > Creating loading time **API** using **Flask**

Loading Time Project : Loading Time API

Working process of Loading Time API:

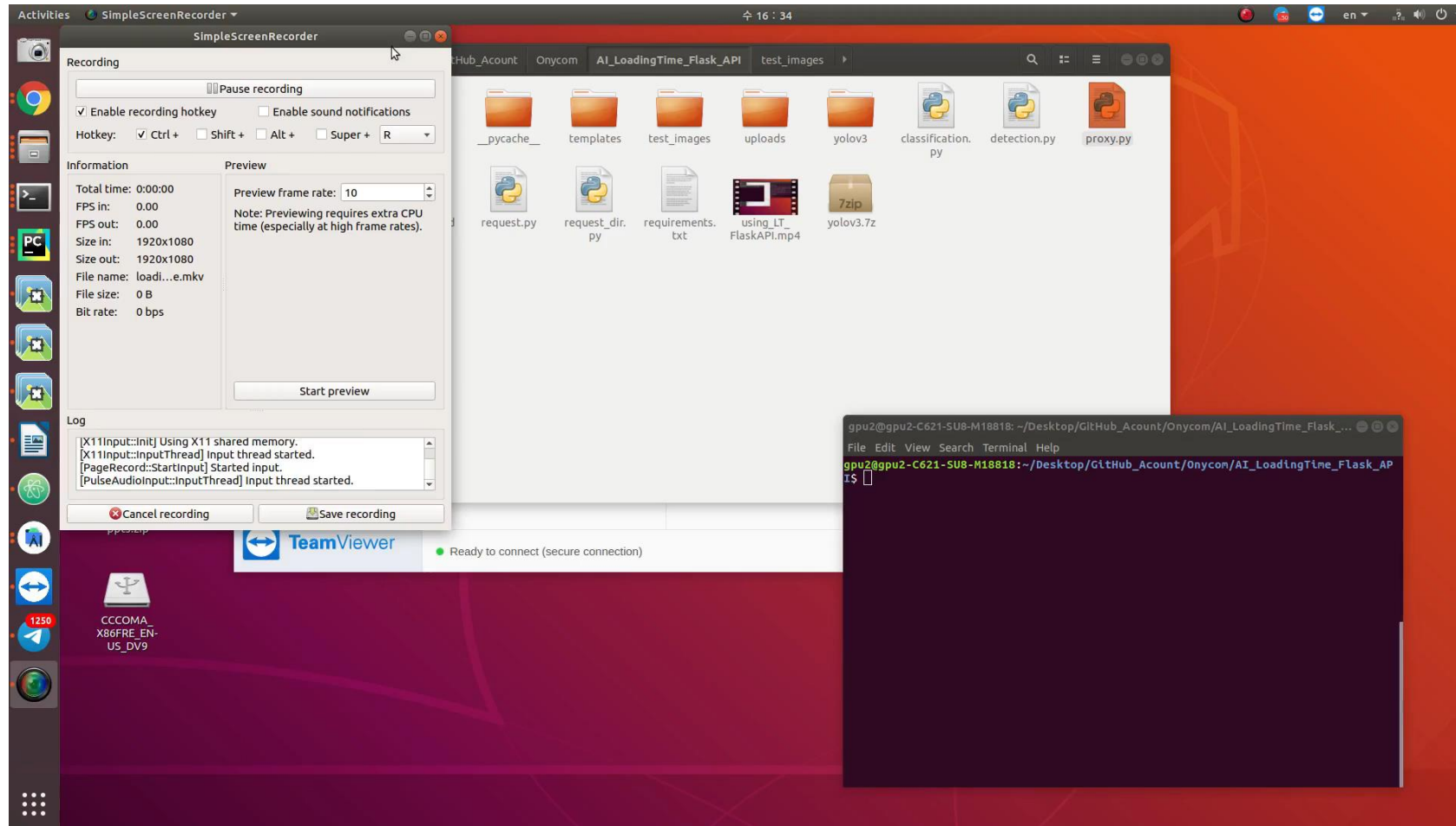


In the above architecture: Image will be sent to YOLOv4, if the response from YOLOv4 is '**loading**', *result is loading*.

If the response from YOLOv4 is '**normal**', *the image* will be sent to **InceptionV3**. **InceptionV3** gives the final result.

Loading Time Project : Loading Time API

Please watch below video to know how **Loading Time API** works:



Play:

Creating Automation Testing Tool for mobile applications

I tried to create **Automation Testing Tool** which is used for *testing mobile application automatically*.

To create Automation Testing Tool, I combined **DroidBot** (link: <https://github.com/honeynet/droidbot>) and **Loading Time API**

Responsibilities of **Automation Testing Tool**:

- Installing mobile app

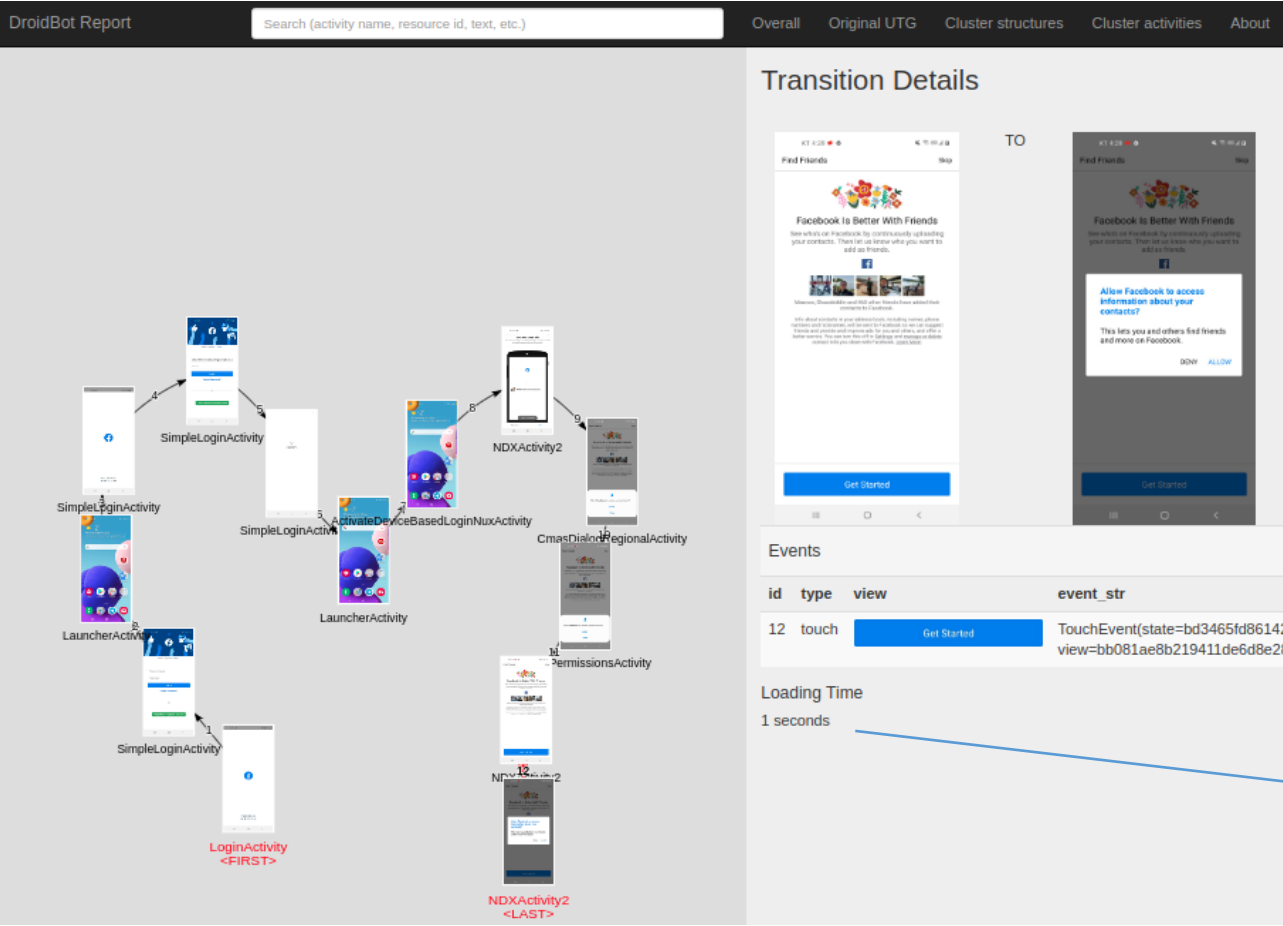
- Touching all clickable objects (image, icon, ...)

- Finding total loading time (Connecting with Loading Time API)

- Showing all result after testing mobile application

Creating Automation Testing Tool for mobile applications

Output result of Automation Tool



Output result of “facebook.apk” after testing by **DroidBot** and **Loading Time API**

State 11 to State 12

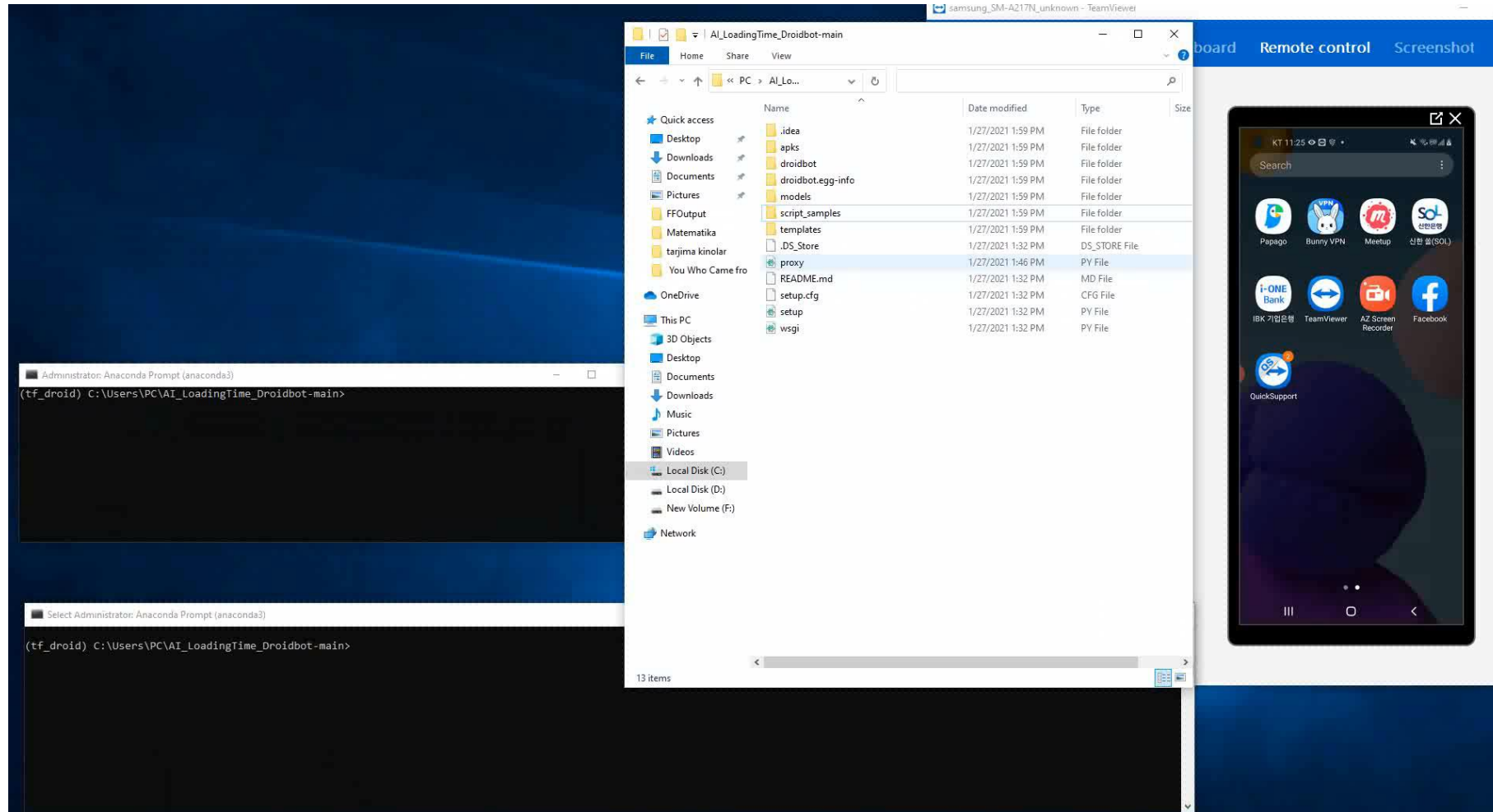
Result = 1 seconds

Result means that Loading time API found only *one loading image* from **State 11** to **State 12**

Creating Automation Testing Tool for mobile applications

Please watch video to see how **Automation Testing Tool** works :

Instagram is used for testing



Detecting clickable objects in mobile applications

Step 1 => **Collecting dataset**: Take screenshots using **150** mobile apps

Step 2 => **Labeling dataset** using **Labellmg** tool (link: <https://github.com/tzutalin/labellmg>)

class names = [icon_button, image, input_box, clickable_item, button, text_button]

dataset examples:



Detecting clickable objects in mobile applications

Step 1 => **Collecting dataset**: Take screenshots using **150** mobile apps

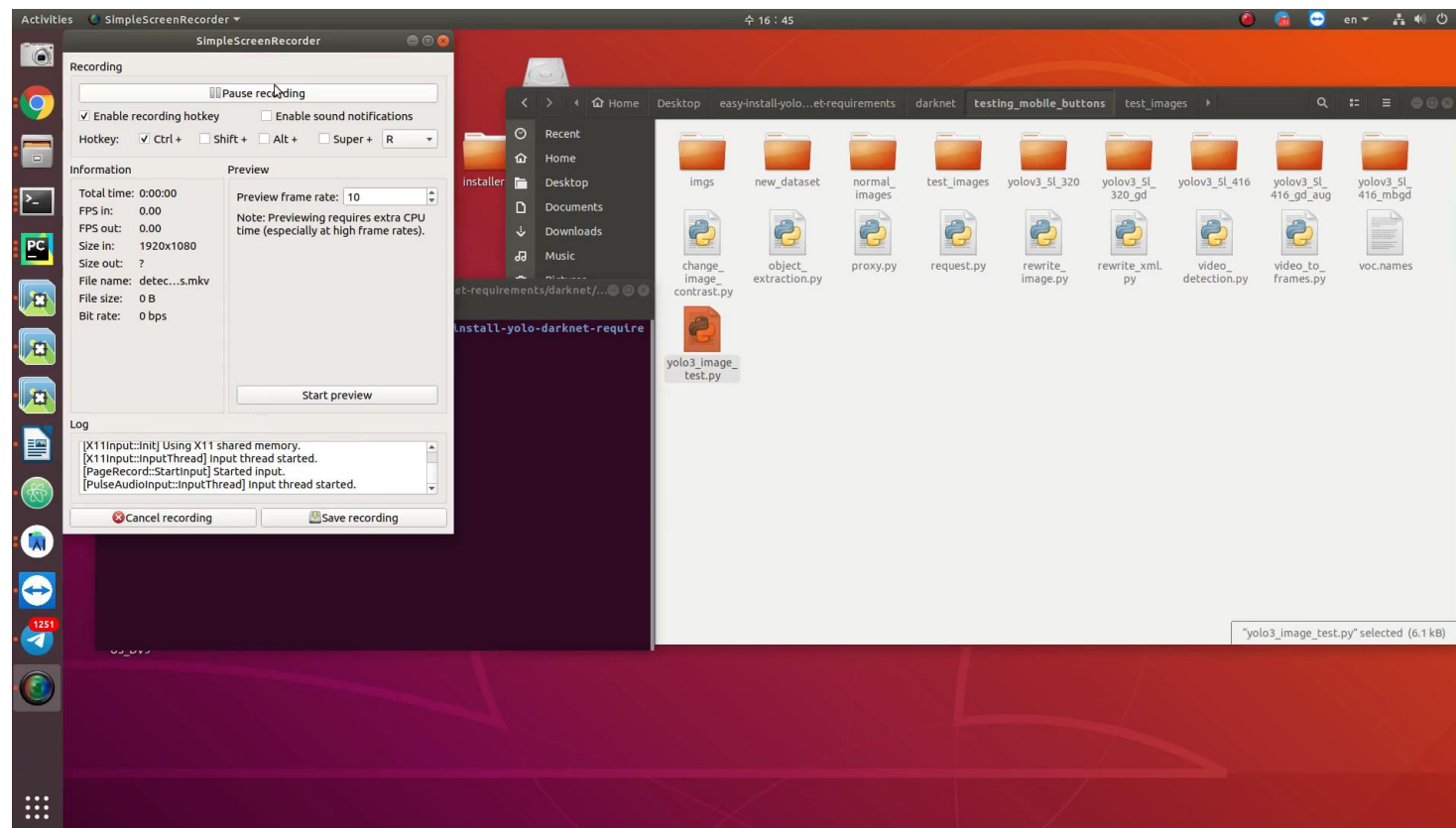
Step 2 => **Labeling dataset** using **Labellmg** tool (link: <https://github.com/tzutalin/labellmg>)

Step 3 => **Making detection model**:

Detection model: Training dataset with **Yolov3** and **Yolov4** using Darknet (link: <https://github.com/AlexeyAB/darknet>)

Step 4 => Creating loading time **API** using **Flask**

Please play the video to see
the result of **Detection Model**:



Creating **Auto-labeling tool** for labeling all objects (buttons, text buttons, image and etc..) on Image

The **goal** of this task is to *label all objects* (with **their names**) on images.

To creating **Auto-labeling**:

Auto-labeling Tool **1** (ALT1) = UI Automator + OCR + Icon classification model

Auto-labeling Tool **2** (ALT2) = Detection Model + OCR + Icon classification model

UI Automator: <https://github.com/openatx/uiautomator2>

Detection Model: which is explained previous pages

OCR: **Easy OCR** open-source project (link: <https://github.com/JaidedAI/EasyOCR>)

Icon classification model: dataset is download from **Kaggle** and dataset was training by **VGG** model

Creating **Auto-labeling tool** for labeling all objects (buttons, text buttons, image and etc..) on Image

input



ALT1 output

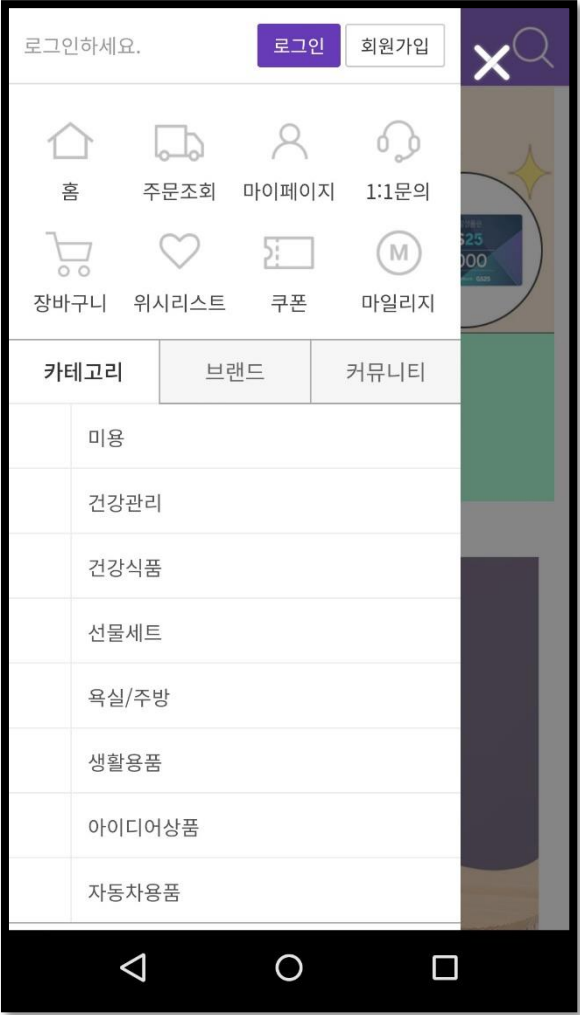


ALT2 output

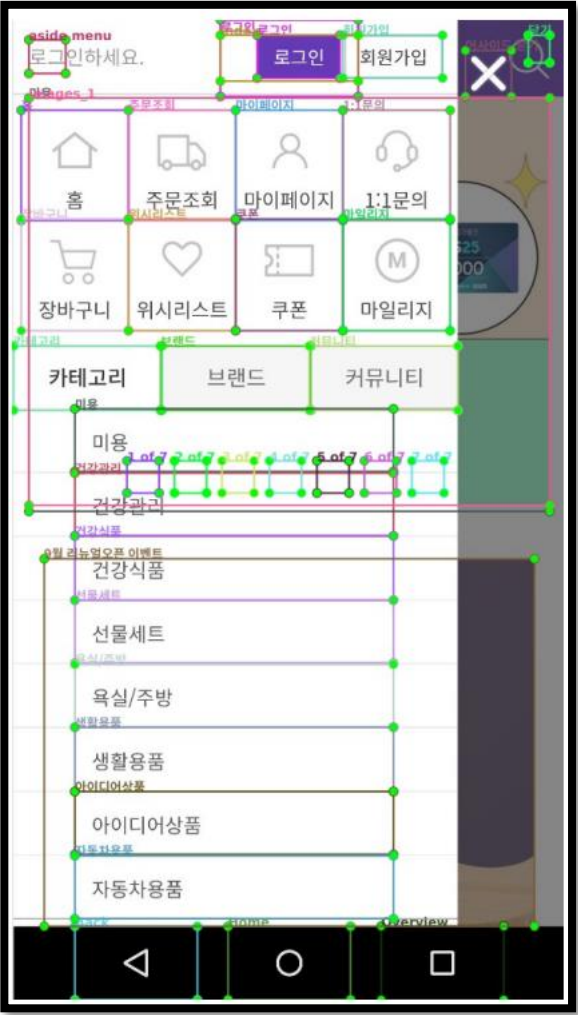


Creating **Auto-labeling tool** for labeling all objects (buttons, text buttons, image and etc..) on Image

input



ALT1 output



ALT2 output



Creating **Auto-labeling tool** for labeling all objects (buttons, text buttons, image and etc..) on Image

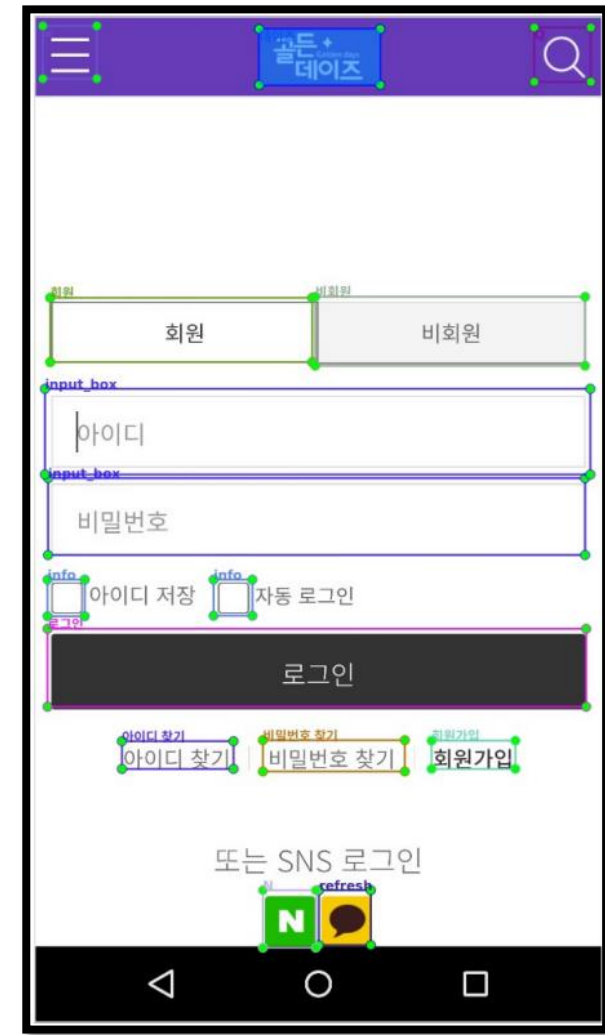
input



ALT1 output



ALT2 output



Multi-devices controlling in the same time

The **goal** of this task is to **control multiple devices in the same time**.

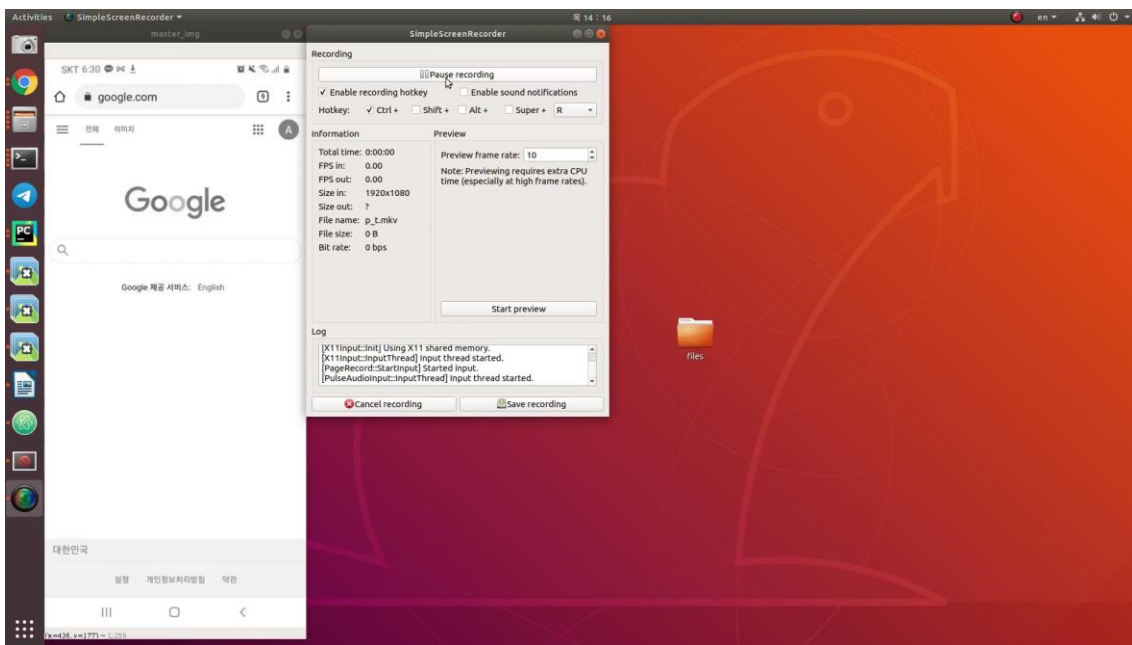
To implement this task, I used “*Multi-scale template matching*” open-source code (link: <https://github.com/agiledots/multiscale-template-matching>),

But I have created **new formula** which help to **decrease the matching time**.

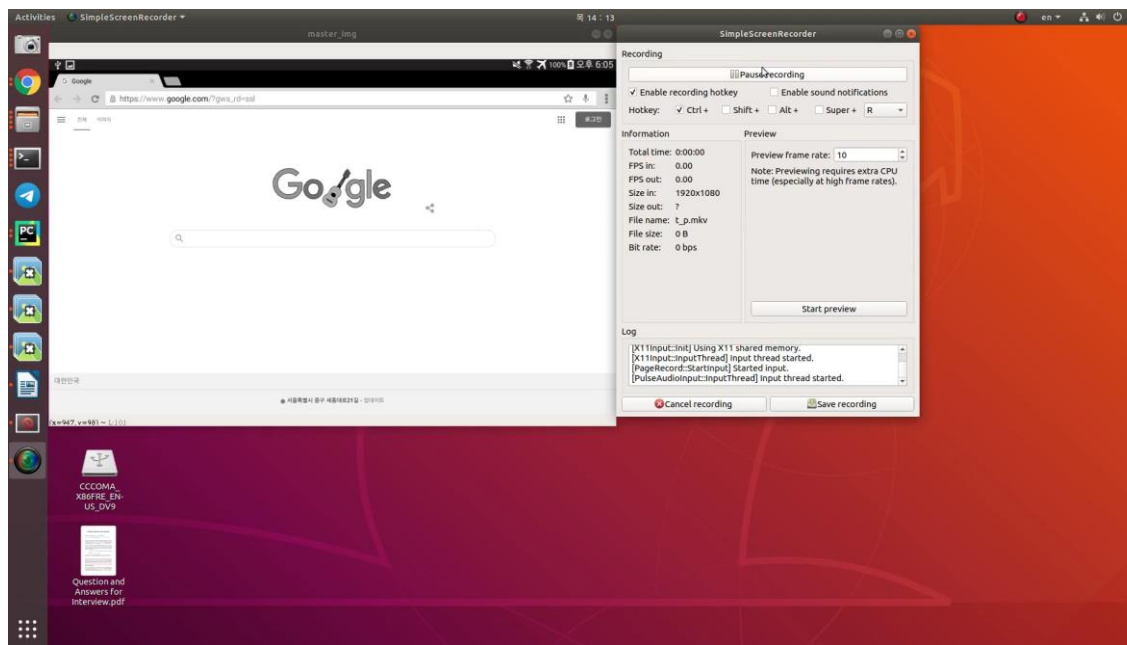
I have implemented this task in **Python** and **C++** .

Please watch below videos for knowing what is **multi-device controlling** and how it works:

Phone with other devices



Tablet with other devices



AI Researches

Related to Face Recognition by using FaceNet
Image Enhancement by GANs and Computer Vision

Using Python with Tensorflow and Keras

AI Project

Café cleanliness using Transfer Learning

Inception, Mobile Net and Efficient Nets [B0, B7]
Best result on Efficient Net B3 and Efficient Net B6

Unoccupied + Empty



Unoccupied + not_Empty



Occupied + not_Empty



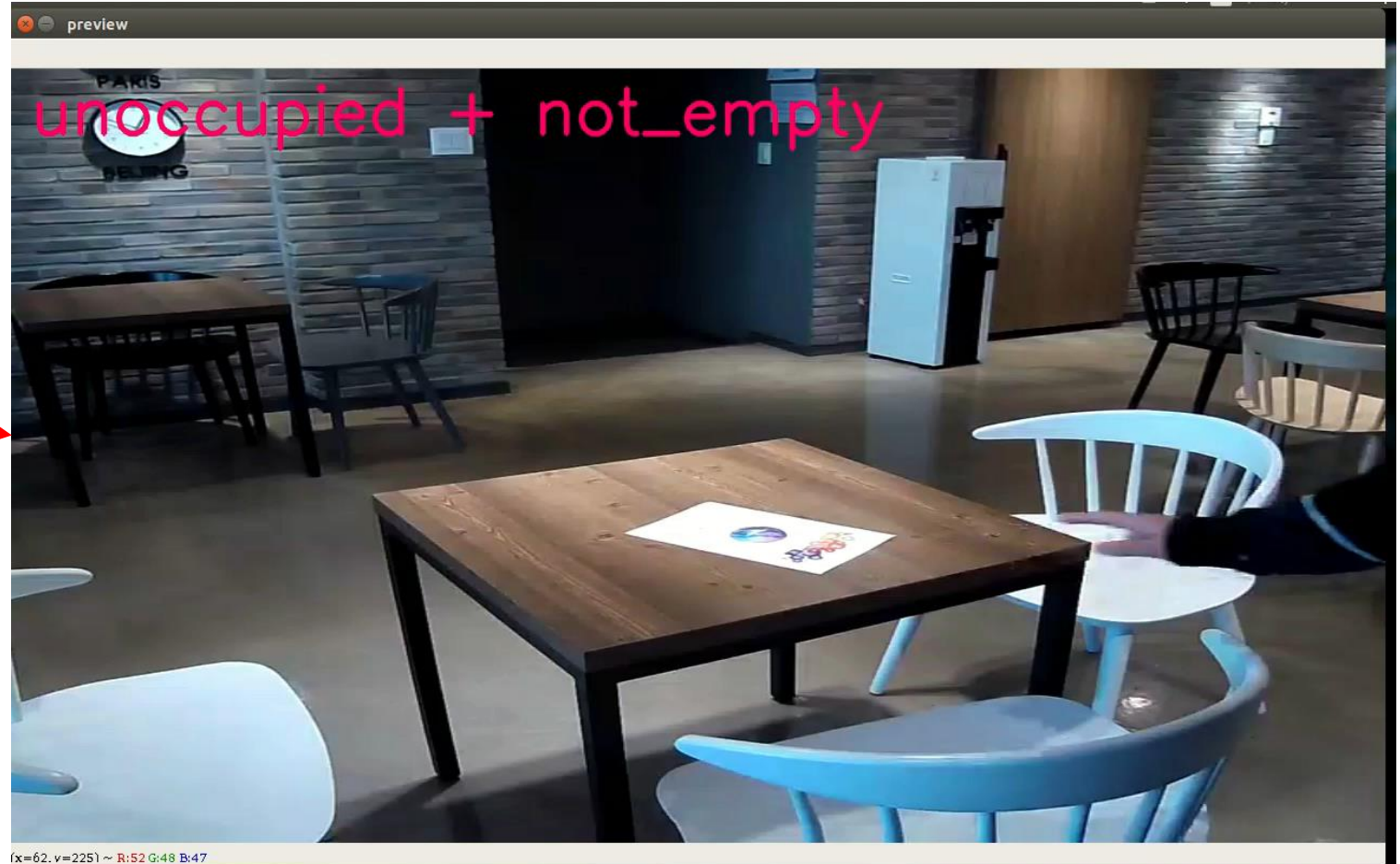
Occupied + Empty



The goal of this Project is :
Improving the quality of service.

Please watch a video

It is real-time testing by using a Camera.
This model always warn the waiter or Café
or Restaurant employees about the table is
empty or not, or a customer came or not



GUI projects

Creating Self-Ordering applications to sell **Coffee, Drinks, and Tea**

Those GUI applications are written for the KIOSK Machine as bellow images:

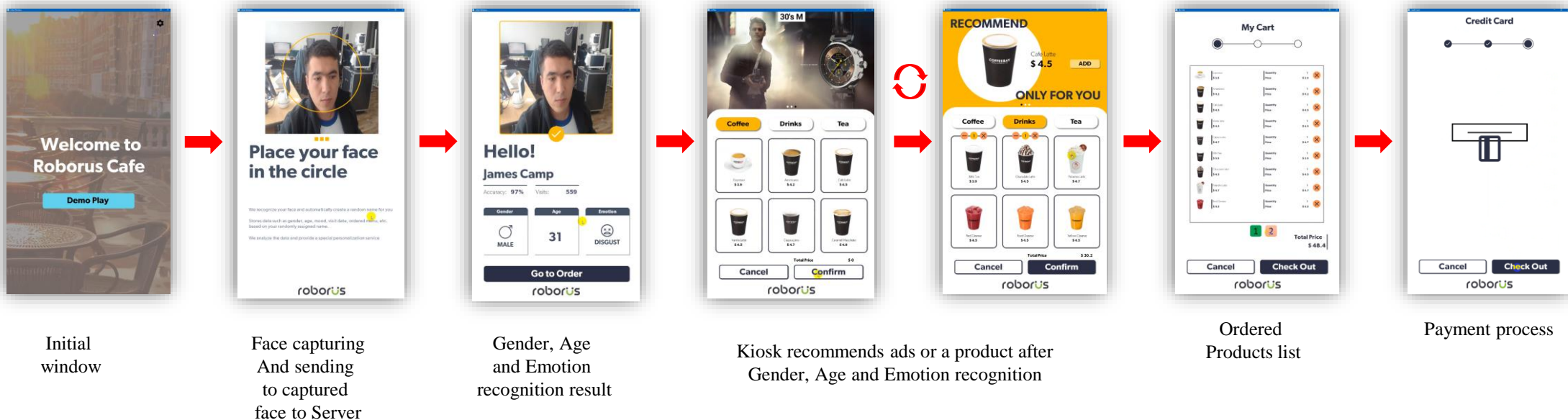
Using C++ with Openframeworks



UI projects

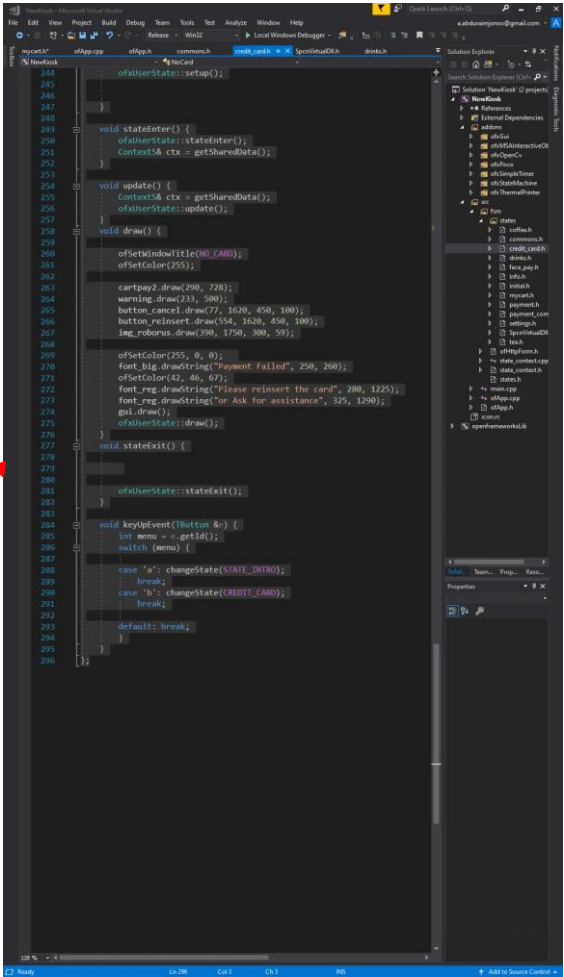
I have created two types of self-ordering applications. The first type of application has a face capturing window. It is very interesting for customers. Because, Kiosk recommends a product and ads after recognizing **Gender, Age, and Emotion**

Following images are **steps** of the first type of self-ordering application



Please watch the videos on the next page, then you totally understand the working process of this application

Please play and watch the video to understand the working process of self-ordering application after Gender, Age, and Emotion recognition



The screenshot shows a code editor with C++ code for a self-ordering application. The code includes state management, drawing functions, and a keypress event handler. A red arrow points from the text box to the code.

```
244 ofUserState::setUp();
245
246
247
248
249 void stateEnter() {
250     ofUserState::stateEnter();
251     ContextSA ctx = getSharedData();
252 }
253
254
255 void update() {
256     ContextSA ctx = getSharedData();
257     ofUserState::update();
258 }
259
260 void draw() {
261     ofSetWindowTitle("ROBO_CARD");
262     ofSetColor(255);
263     cartpay2.draw(200, 720);
264     warning.draw(233, 500);
265     button_cancel.draw(77, 560, 450, 100);
266     button_reinsert.draw(554, 1620, 450, 100);
267     img_roborus.draw(200, 1750, 300, 593);
268
269     ofSetColor(255, 0, 0);
270     font_big.drawString("Payment failed", 250, 260);
271     ofSetColor(42, 46, 67);
272     font_reg.drawString("Please reinsert the card", 200, 1225);
273     font_reg.drawString("or Ask for assistance", 325, 1290);
274     gui.draw();
275     ofUserState::draw();
276 }
277
278 void stateExit() {
279
280 }
281
282 ofUserState::stateExit();
283
284
285 void keyPressEvent(Button &) {
286     int menu = i.getId();
287     switch (menu) {
288     case 'a': changeState(STATE_INTR0);
289         break;
290     case 'b': changeState(CH001_CARD);
291         break;
292     default: break;
293 }
294
295 }
296
```

GUI projects

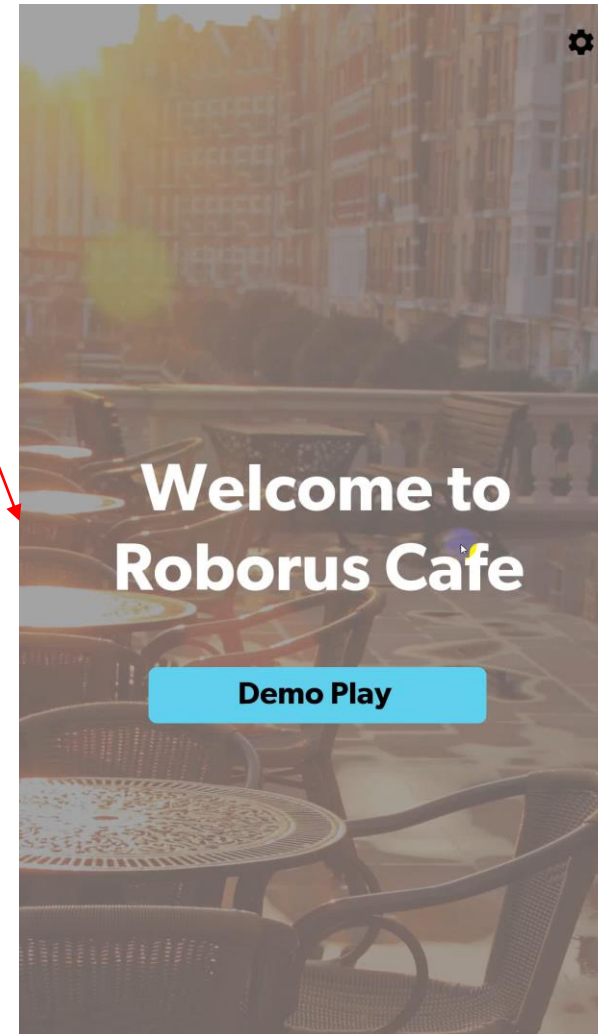
The second type of application has no Gender, Age, and Emotion recognition part. Customers can order without Face capturing.

Following images are **steps** of the second type of self-ordering application



Please watch the videos on the next page, then you totally understand the working process of this application

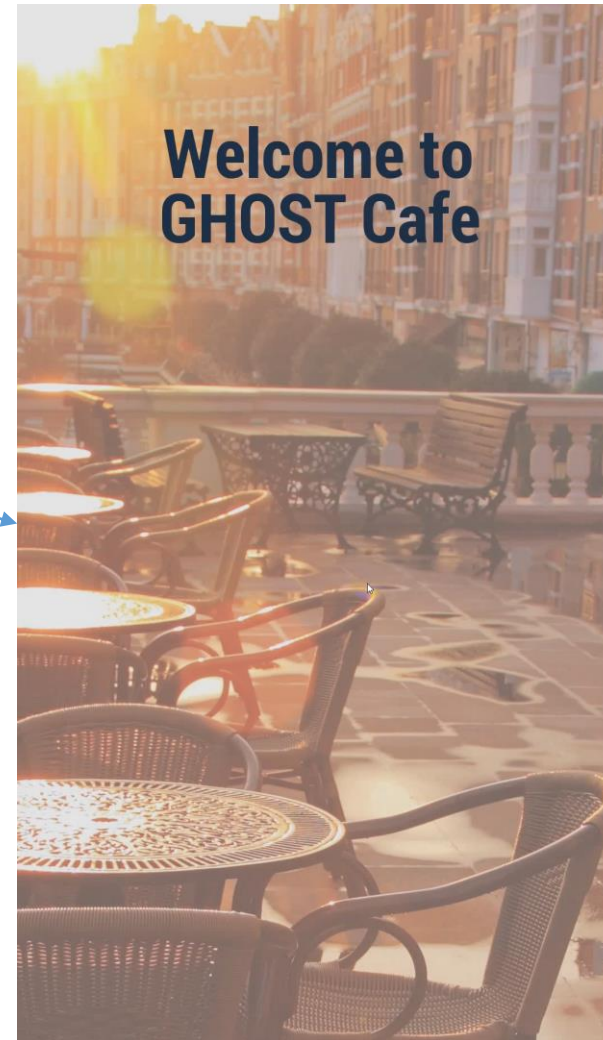
Please play and watch the video to understand the working process of self-ordering application without face capturing



Similar to the second type of self-ordering application.

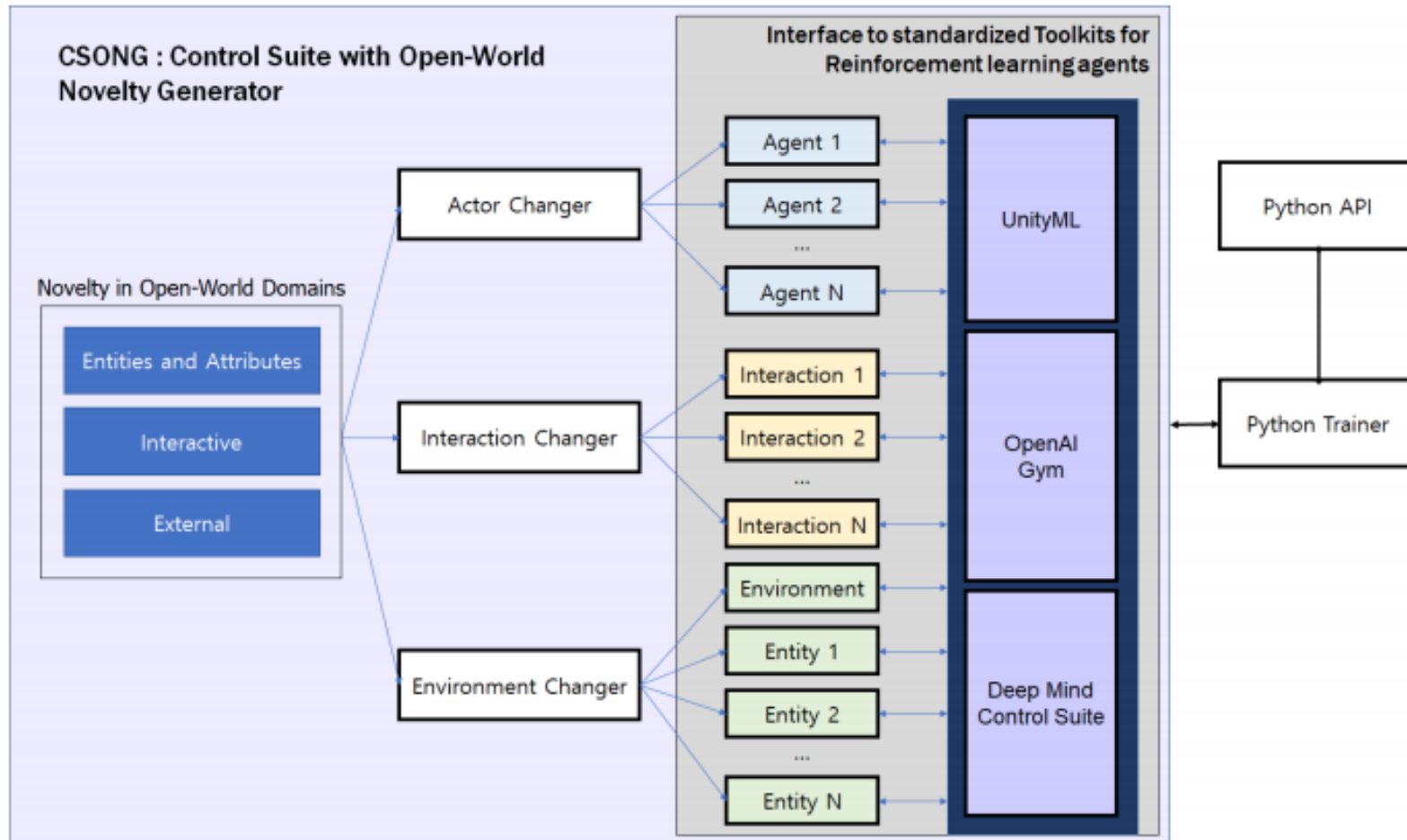
However, the design is different from the previous one.

Please, watch following video



CSONG Development

CSONG : RL **Control Suite with Open-World Novelty Generator**



CSONG Development : Server side of CSONG

Novelty Authoring Tool for CSONG : Control Suite with Open-World Novelty Generator

Domain Selector

Mujoco Domain
Ant

Domain Editor

State Space

Unique ID Leg10
Total Type Count 2 Dimension : 6
Type 1 Position(x, y, z)
Type 2 Angular Velocity(Vx, Vy, Vz)

Action Space

Joint# Leg10
Total Action Types 2
Action Type 1
Action Type 2

Other Parameters

Ant xml
Legs#

Relation Editor

Property of Objects

Replacement Shortcut : Ctrl + Num 1
Object Property 1 A:WInteractionBehaviorWfriendly-soft.cs
Object Property 2 A:WInteractionBehaviorWfriendly-strict.cs
Object Property 3 A:WInteractionBehaviorWadversarial-soft.cs
Object Property 4 A:WInteractionBehaviorWadversarial-strict.cs

Reward of Actions

Replacement Shortcut : Ctrl + Num 1
Reward 1 A:WRewardsOfActionsWReward1.cs
Reward 2 A:WRewardsOfActionsWReward2.cs
Reward 3 A:WRewardsOfActionsWReward3.cs
Reward 4 A:WRewardsOfActionsWReward4.cs

Environment Editor

Environment Change

Ground Size : Width 100 Length 100 Tiled?
Tile Size : X 100 Y 100
Physics Engine : ☒ PhysicsX ☐ pyBullet ☐ MJFC

Entities(Obstacles)

Total Entity Count : 3
Entity 3D Model : A:WModelWObstacle1.fbx
Entity 1 : Pos X 0 Y 0 Z 0 Vel X 0 Y 0 Z 0
Entity 2 : Pos X 10 Y 10 Z 10 Vel X 0 Y 0 Z 0
Entity 3 : Pos X 0 Y 0 Z 10 Vel X 0 Y 0 Z 0

Goal Change

Replacement Shortcut : Ctrl + Num 2
Goal Template 1 A:WGoalsWgoal1.cs
Goal Template 2 A:WGoalsWgoal2.cs
Goal Template 3 A:WGoalsWgoal3.cs
Goal Template 4 A:WGoalsWgoal4.cs

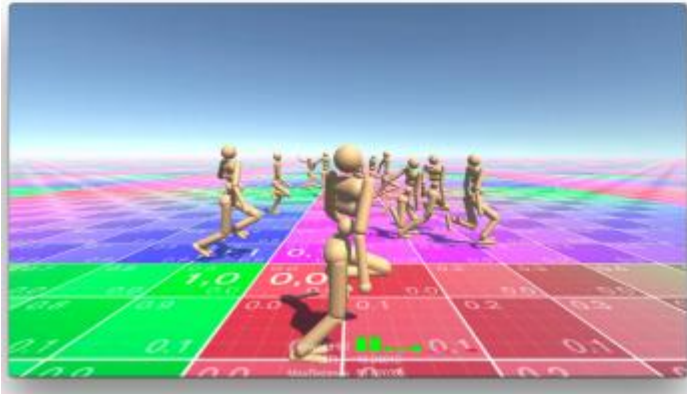
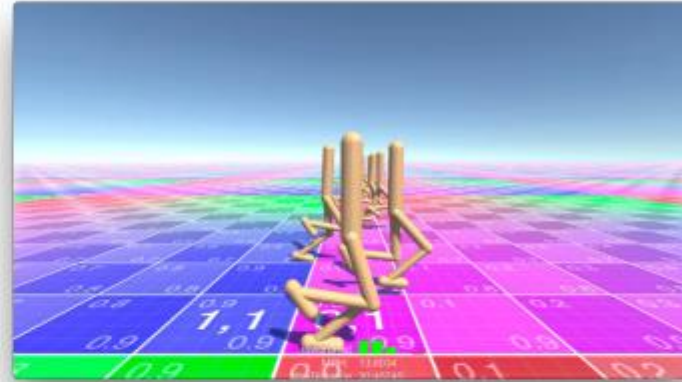
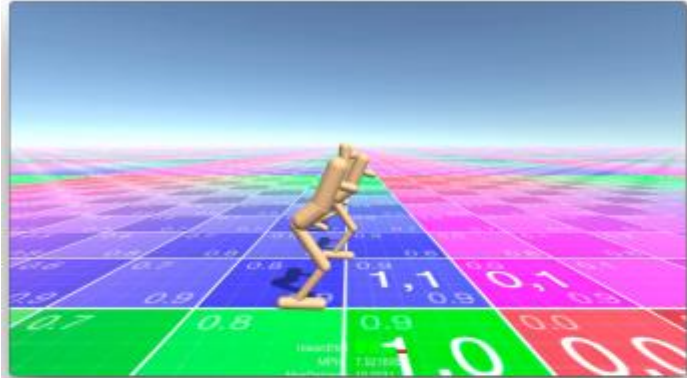
Reset

Apply Changes

Quit

Server Side is Windows Form Application by C# (.NET Framework)

CSONG Development : Client side of CSONG



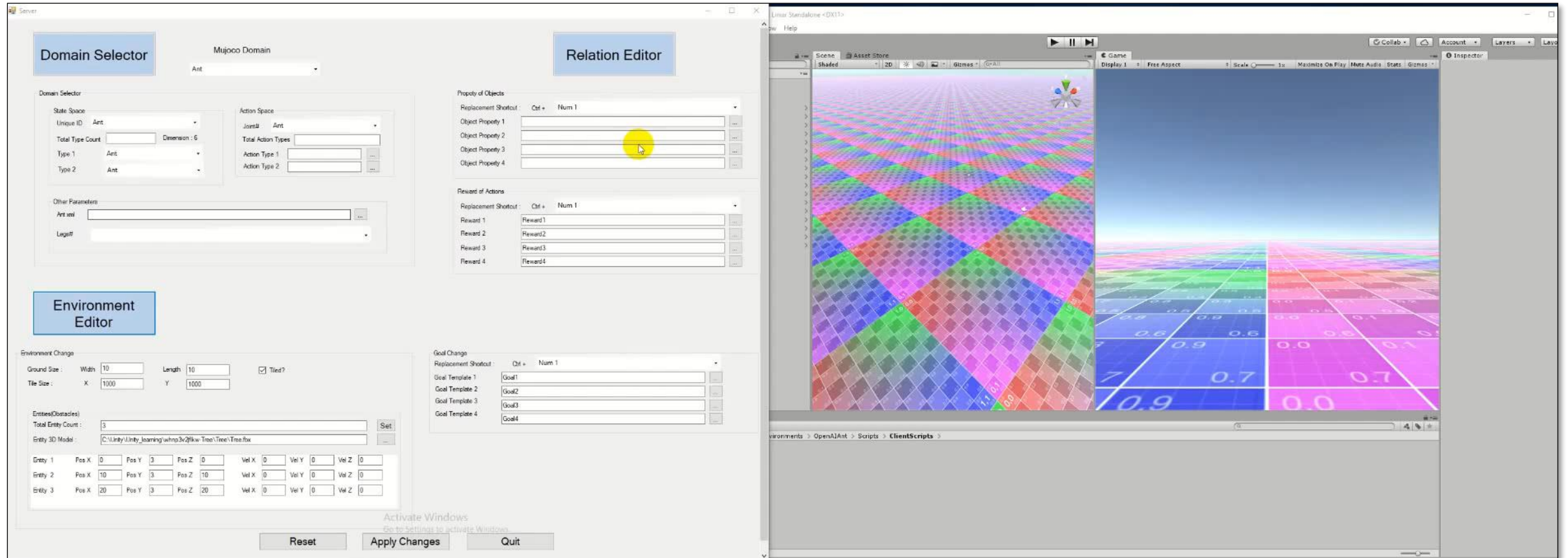
I implemented CSONG for Ant Model.

In future, CSONG will be Implemented for all Models

Unity ML-Agents. The Marathon Environments

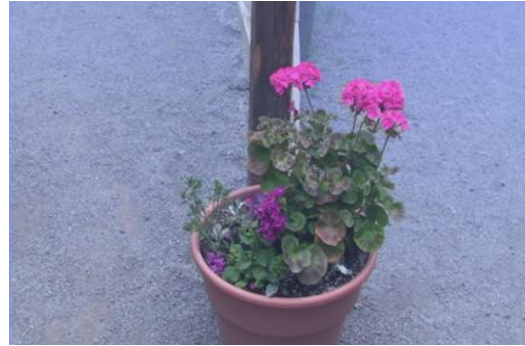
CSONG Development : Final Result

Please watch below video:



Computer Vision Researches

Increasing the quality of image using “Non-local image dehazing” algorithm and improving it

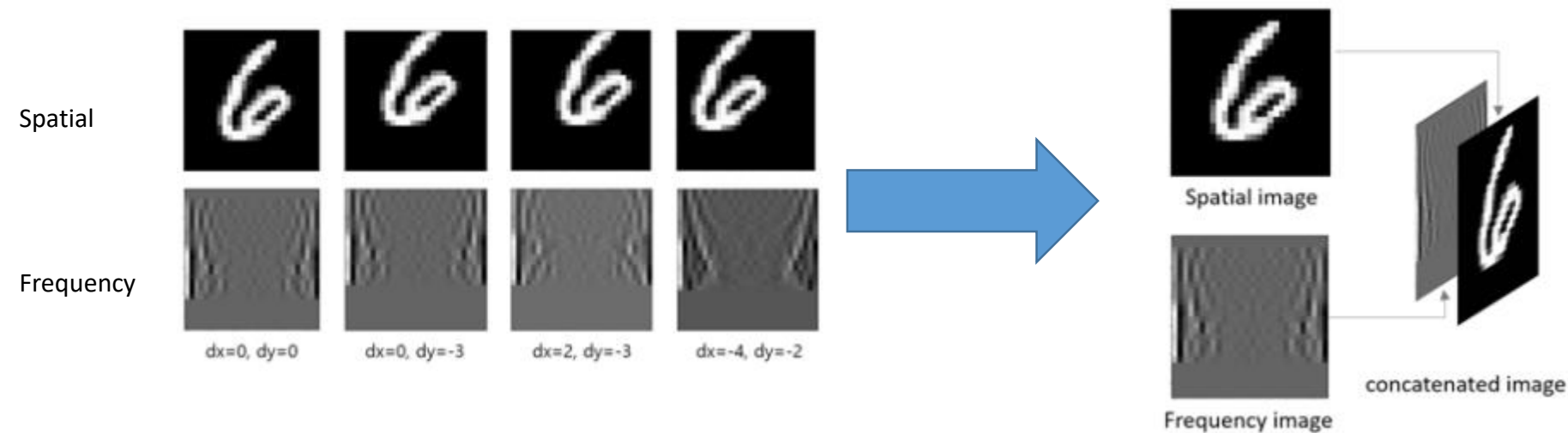


AI Researches and Paper summary

Paper Name: Extending Input Channel Using Global Feature Image for Convolutional Neural Networks

Paper link : <https://ieeexplore.ieee.org/document/8960966>

Short explanation about paper: All digits are centralized in MNIST. However, if the digit is shifted from the center, the accuracy will decrease. **Fast Fourier Transform** (FFT) can produce similar frequency image from shifted images.



AI Researches and Paper summary

Paper Name: Extending Input Channel Using Global Feature Image for Convolutional Neural Networks

Paper link : <https://ieeexplore.ieee.org/document/8960966>

Result: The following table is the accuracy comparison result among spatial, frequency, and concatenated image. You can see, mostly the concatenated achieve higher accuracy than others

Translation (dx, dy)	Spatial image	Frequency image	Concatenated image
0,0	99.53	97.72	99.51
0,-3	93.17	84.63	94.37
2,-3	92.13	79.60	93.51
4,0	92.15	76.39	93.42
2,3	91.21	74.51	93.39
0,3	94.66	86.33	94.95
-4,3	88.77	60.26	91.07
-5,0	85.20	61.16	88.89
-4,2	82.75	60.43	88.67
-3,-2	91.54	71.37	94.37
Total	91.11	75.24	93.21
deviation	4.71	12.44	3.15

Hello Dears:

First of all thank you very much for reading my long portfolio. I am really sorry if you didn't satisfy my explanations about my researches and projects. However, if you invite me for the interview, I will try to explain all pages more clearly and deeply.

Thank you so much for your attention 🤗