InnoGames

AI FOR GAME INTERACTION

Results from AI Jam 2024

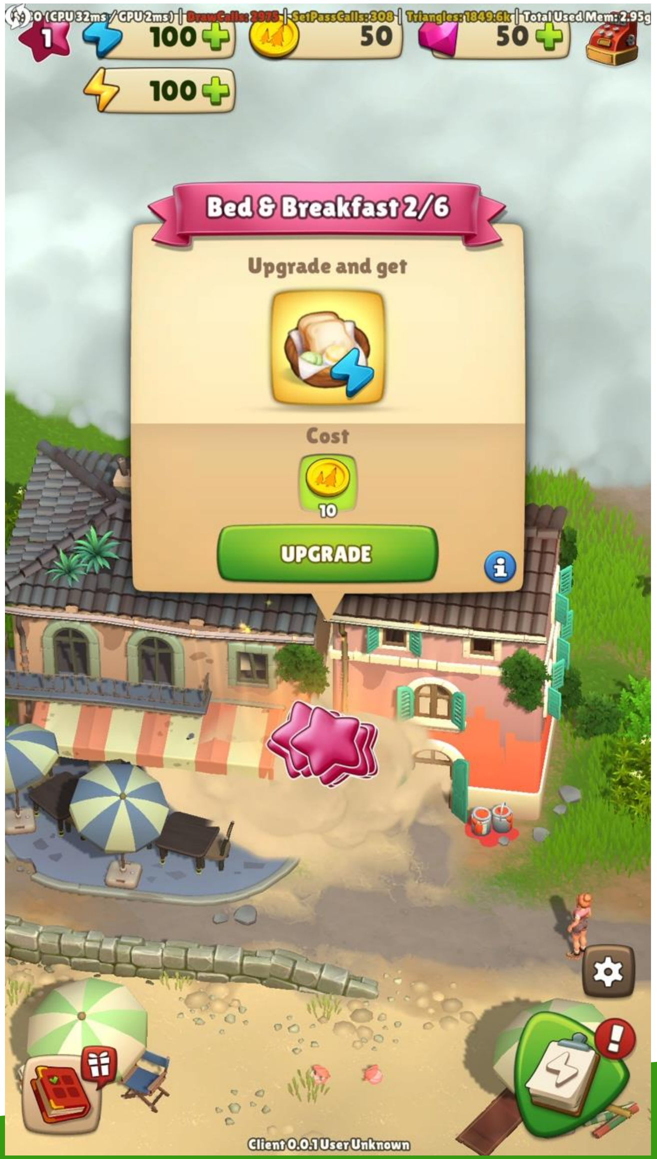
TABLE OF CONTENTS

01

PROBLEM

INTEGRATION TESTS DON'T ADAPT

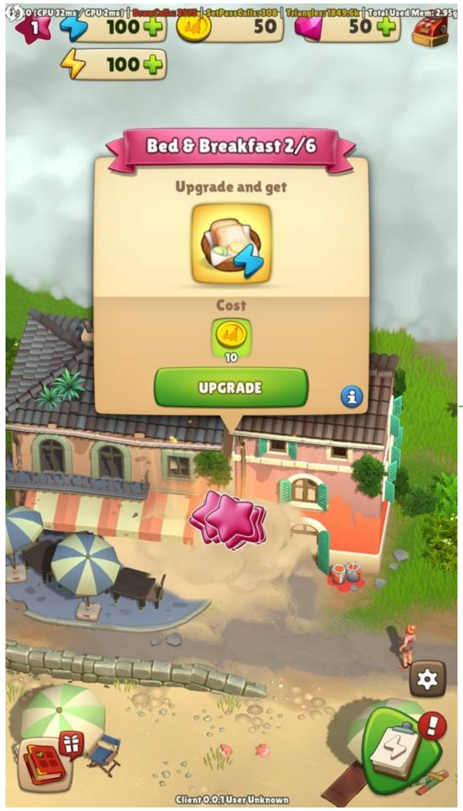
- Integration tests need maintenance once written when something changes in the UI
- Writing integration tests is tedious and time consuming



02

CONCEPT

LET’S USE AI!



1

AI identifies all buttons with their positions on screen

2

AI identifies the button that is meant

3

Position for a click is sent back to Unity

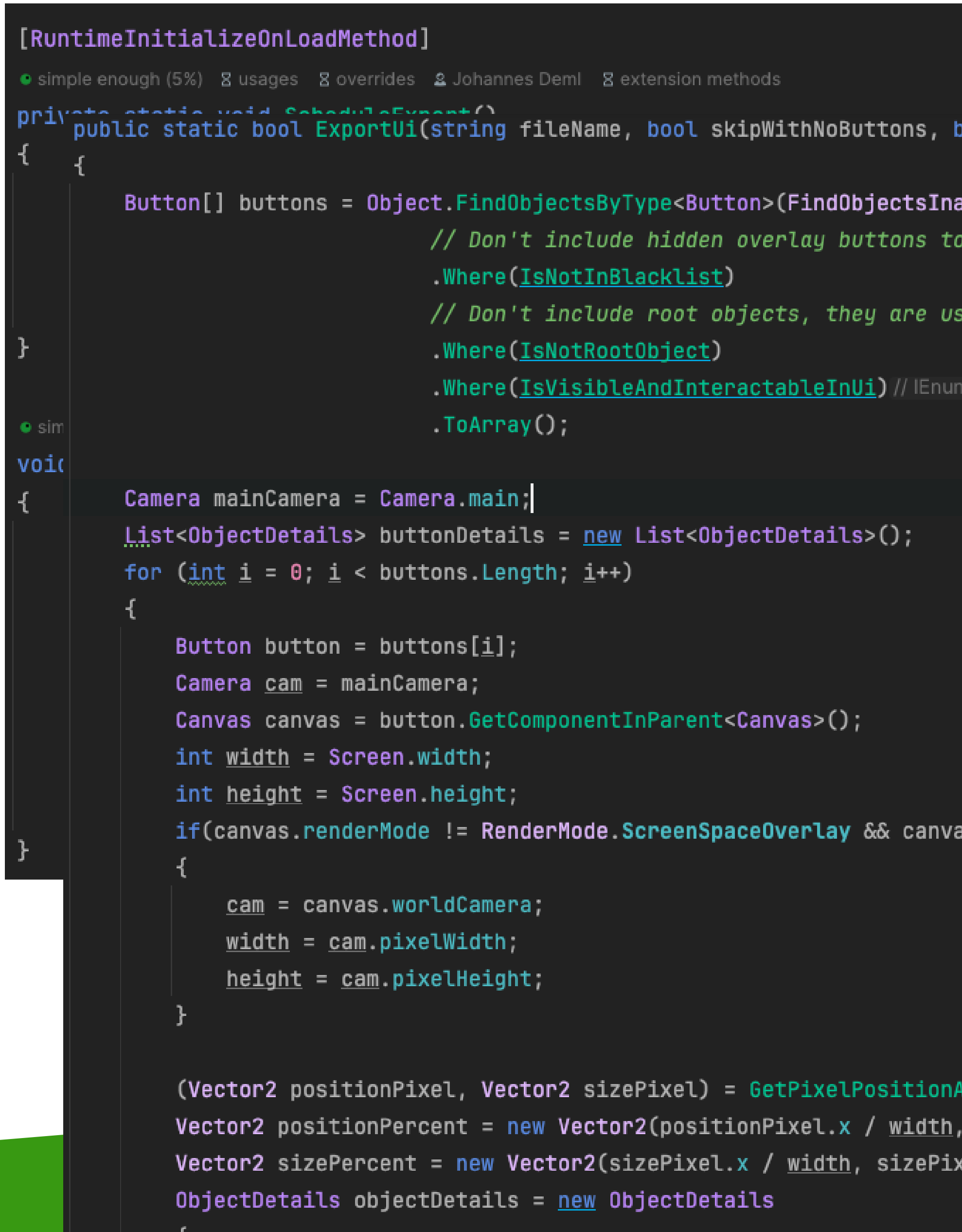
Prompt:
Click on settings button

03

FEEDING DATA

IMAGE CLASSIFICATION FOR BUTTONS

- ChatGPT does not support giving out positions, therefore using YOLOv8
- We need to feed in training data and test data to train the model
- Not much time, manual work would be tedious and we're not sure how much we need to get good results
- We can get all unity buttons on the screen through script
- We have integration tests that already play the game
- Let's take a screenshot every two seconds and save out all valid button positions
- Make sure to throw away all disabled and invisible buttons and other things that are not really buttons (backgrounds)



```
[RuntimeInitializeOnLoadMethod]
simple enough (5%)  usages  overrides  Johannes Deml  extension methods
```

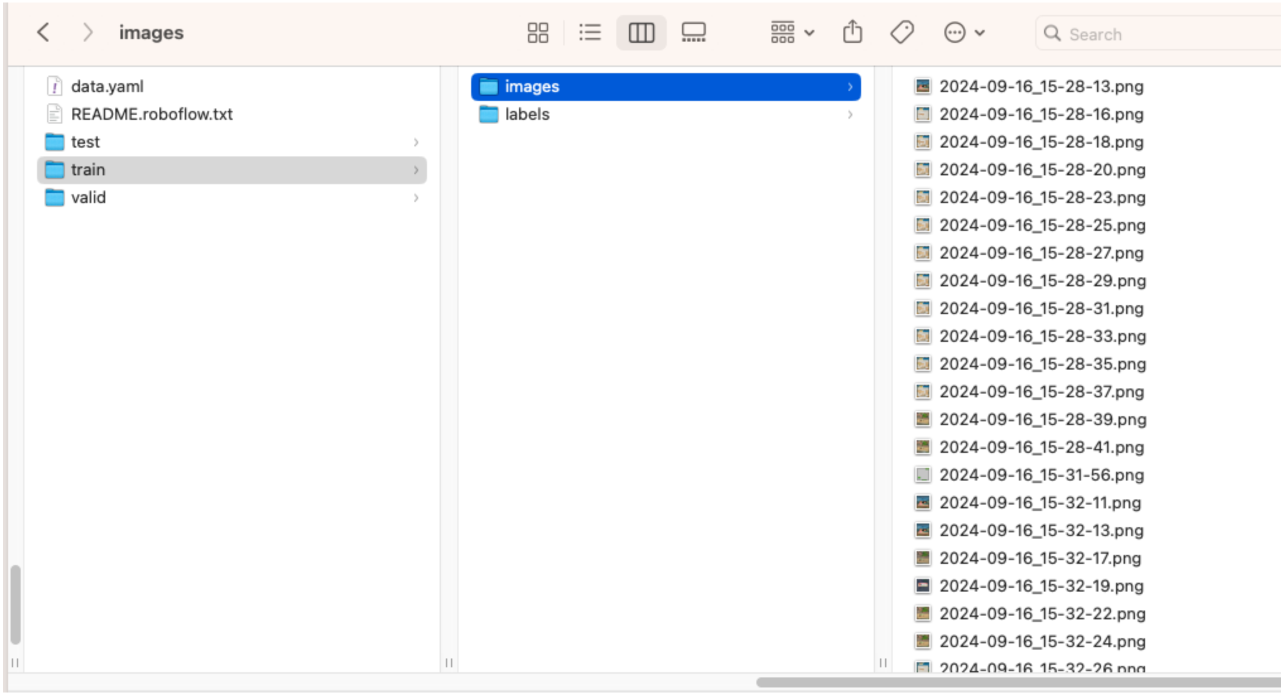
```
public static bool ExportUi(string fileName, bool skipWithNoButtons, b
{
    Button[] buttons = Object.FindObjectsByType<Button>(FindObjectsIna
                                // Don't include hidden overlay buttons to
                                .Where(IsNotInBlacklist)
                                // Don't include root objects, they are us
                                .Where(IsNotRootObject)
                                .Where(IsVisibleAndInteractableInUi) // IEnum
                                .ToArray();

    Camera mainCamera = Camera.main;
    List<ObjectDetails> buttonDetails = new List<ObjectDetails>();
    for (int i = 0; i < buttons.Length; i++)
    {
        Button button = buttons[i];
        Camera cam = mainCamera;
        Canvas canvas = button.GetComponentInParent<Canvas>();
        int width = Screen.width;
        int height = Screen.height;
        if(canvas.renderMode != RenderMode.ScreenSpaceOverlay && canva
        {
            cam = canvas.worldCamera;
            width = cam.pixelWidth;
            height = cam.pixelHeight;
        }

        (Vector2 positionPixel, Vector2 sizePixel) = GetPixelPositionA
        Vector2 positionPercent = new Vector2(positionPixel.x / width,
        Vector2 sizePercent = new Vector2(sizePixel.x / width, sizePix
        ObjectDetails objectDetails = new ObjectDetails
```

IMAGE CLASSIFICATION FOR BUTTONS

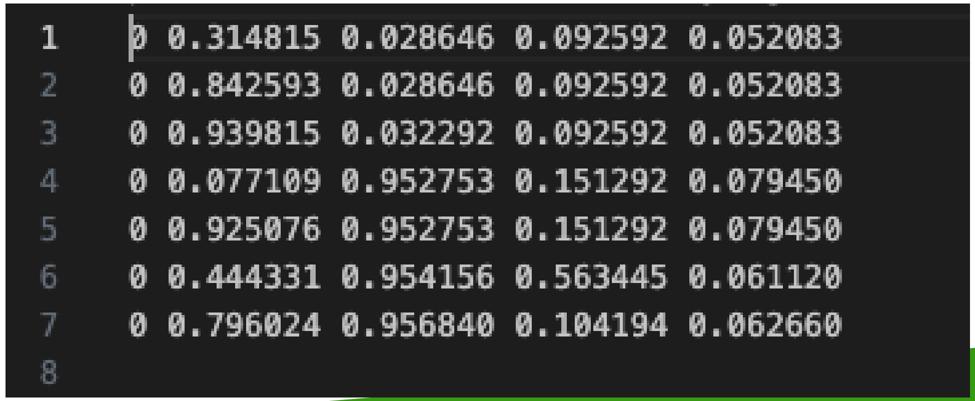
- Next we need to get everything in the right format
- And split them into test and training data
- In our case:
  - test: 1 image
  - train: 82 images
  - valid: 9 images
- Make sure that all images have a size of 640x640
- Label format all in percent
- Train the data



data.yaml
README.roboflow.txt
test
train
valid

images
labels

2024-09-16_15-28-13.png
2024-09-16_15-28-16.png
2024-09-16_15-28-18.png
2024-09-16_15-28-20.png
2024-09-16_15-28-23.png
2024-09-16_15-28-25.png
2024-09-16_15-28-27.png
2024-09-16_15-28-29.png
2024-09-16_15-28-31.png
2024-09-16_15-28-33.png
2024-09-16_15-28-35.png
2024-09-16_15-28-37.png
2024-09-16_15-28-39.png
2024-09-16_15-28-41.png
2024-09-16_15-31-56.png
2024-09-16_15-32-11.png
2024-09-16_15-32-13.png
2024-09-16_15-32-17.png
2024-09-16_15-32-19.png
2024-09-16_15-32-22.png
2024-09-16_15-32-24.png
2024-09-16_15-32-26.png



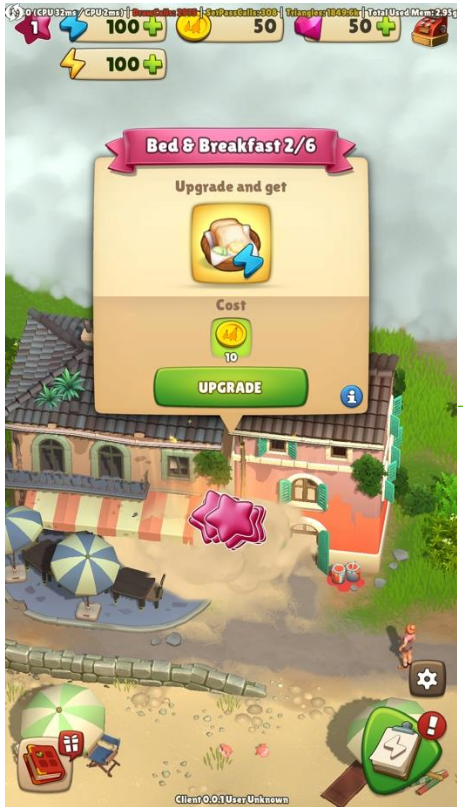
```
0 0.314815 0.028646 0.092592 0.052083
0 0.842593 0.028646 0.092592 0.052083
0 0.939815 0.032292 0.092592 0.052083
0 0.077109 0.952753 0.151292 0.079450
0 0.925076 0.952753 0.151292 0.079450
0 0.444331 0.954156 0.563445 0.061120
0 0.796024 0.956840 0.104194 0.062660
```

04

RESULTS

THE PIPELINE

Firebase Genkit



1

AI identifies all buttons with their positions on screen

YOLOv8

2

AI identifies the button that is meant

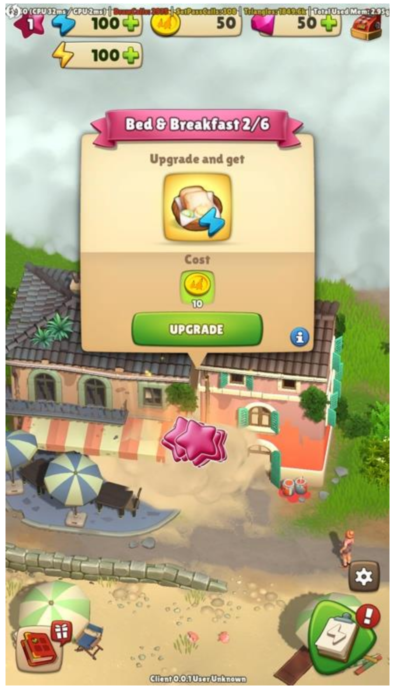
GPT-4o

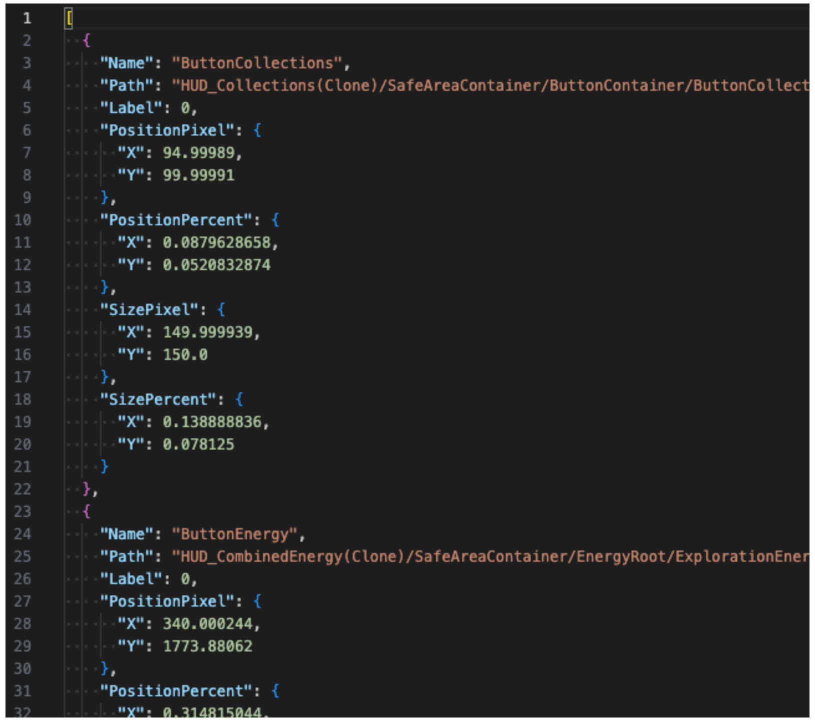
3

Position for a click is sent back to Unity

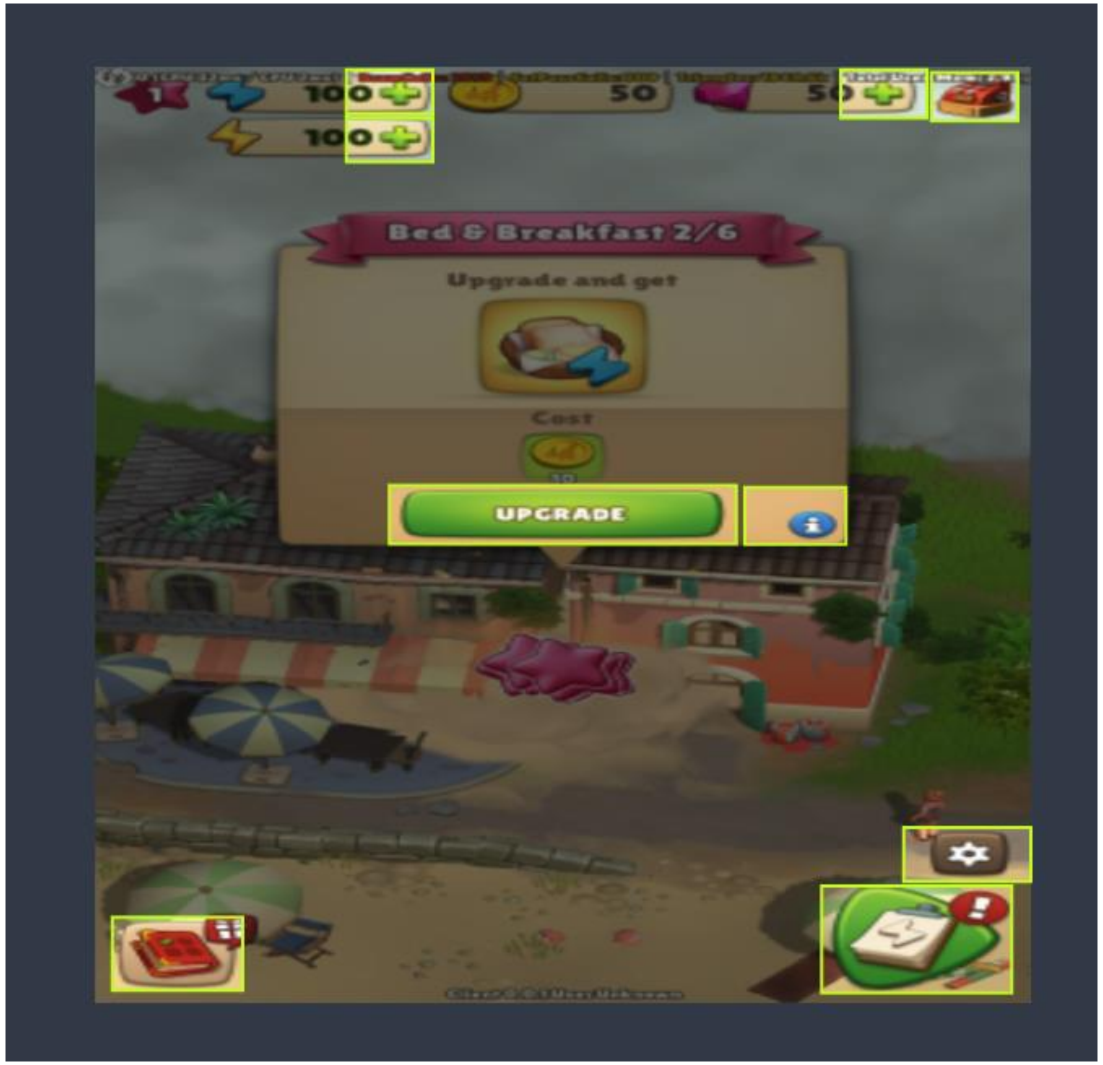
Prompt:
Click on settings button

1: IDENTIFY BUTTONS

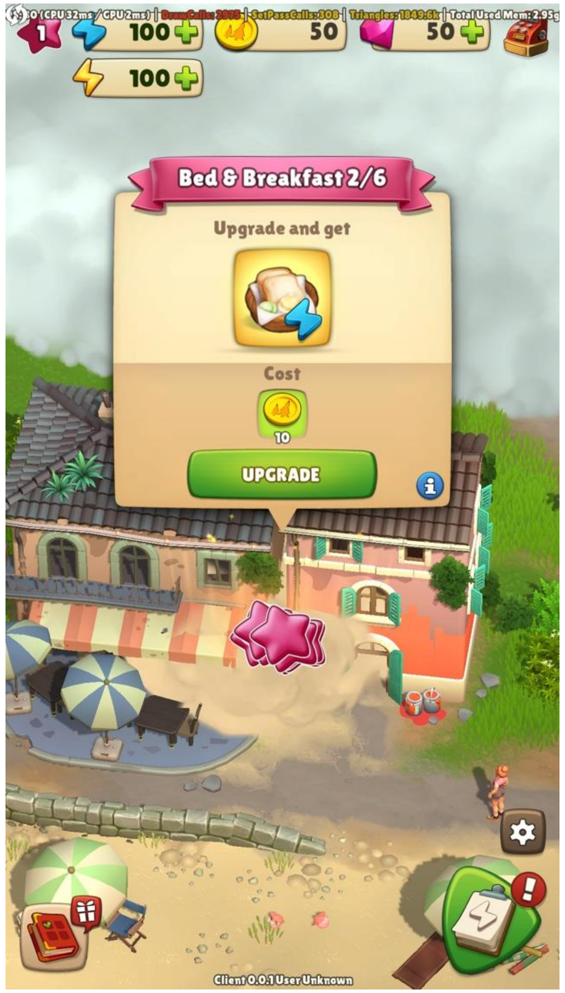




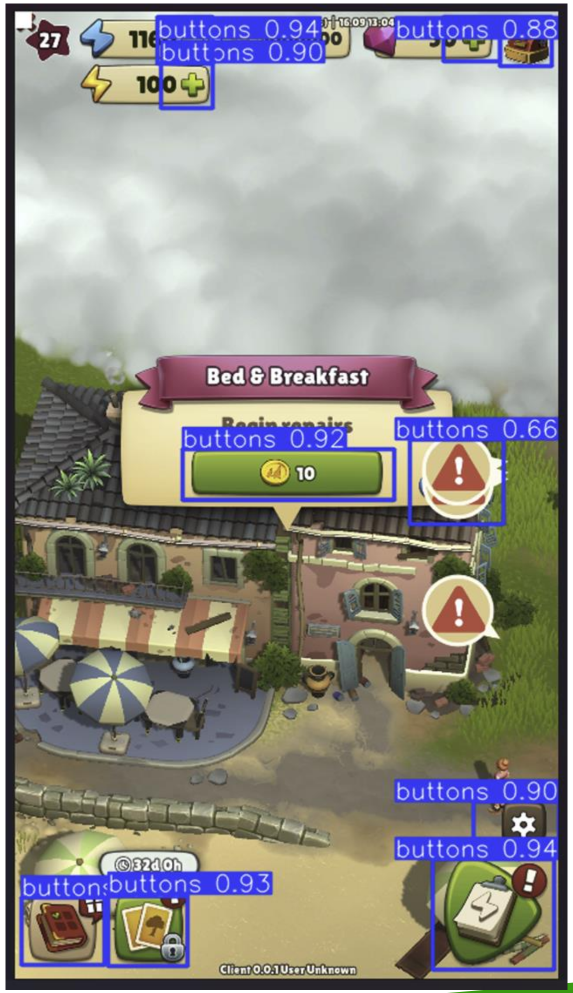
```
[
  {
    "Name": "ButtonCollections",
    "Path": "HUD_Collections(Clone)/SafeAreaContainer/ButtonContainer/ButtonCollect
    "Label": 0,
    "PositionPixel": {
      "X": 94.99989,
      "Y": 99.99991
    },
    "PositionPercent": {
      "X": 0.0879628658,
      "Y": 0.0520832874
    },
    "SizePixel": {
      "X": 149.999939,
      "Y": 150.0
    },
    "SizePercent": {
      "X": 0.138888836,
      "Y": 0.078125
    }
  },
  {
    "Name": "ButtonEnergy",
    "Path": "HUD_CombinedEnergy(Clone)/SafeAreaContainer/EnergyRoot/ExplorationEner
    "Label": 0,
    "PositionPixel": {
      "X": 340.000244,
      "Y": 1773.88062
    },
    "PositionPercent": {
      "X": 0.314815044,
```



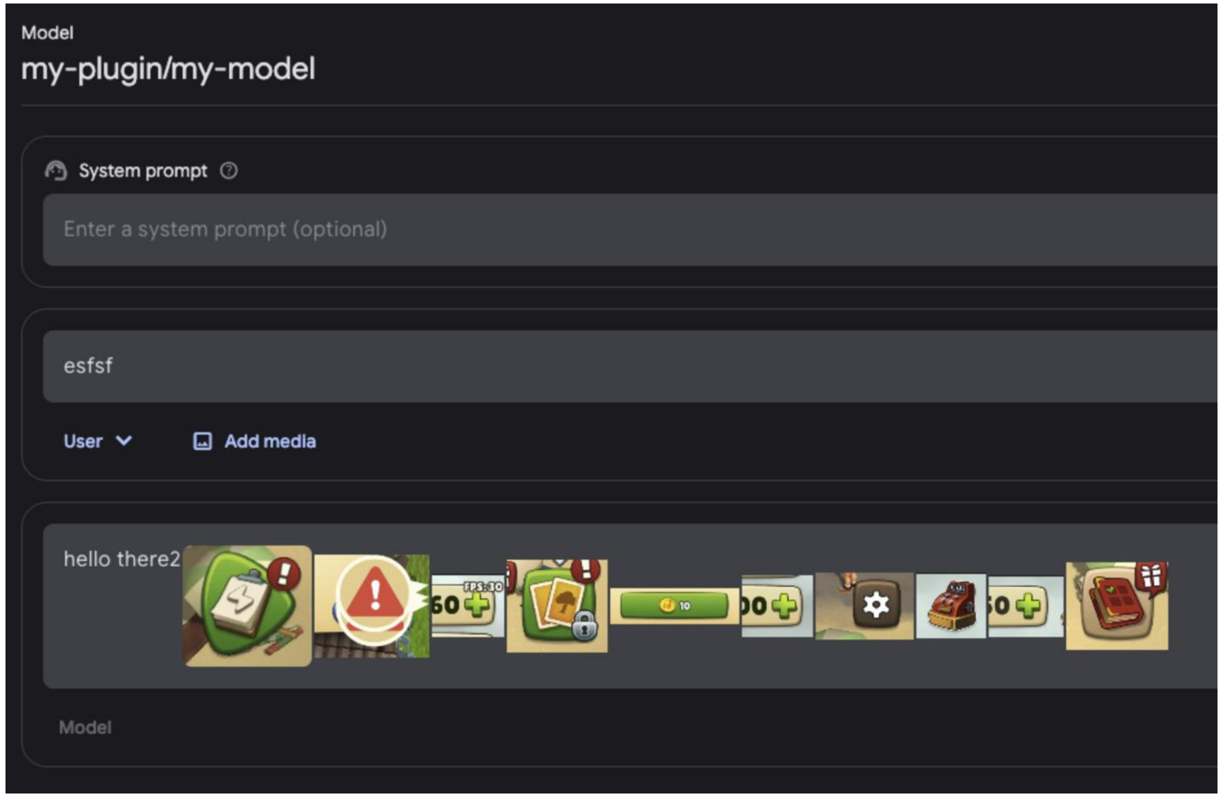
1: IDENTIFY BUTTONS



Trained model



2: CHOOSE CORRECT BUTTON


Model
my-plugin/my-model

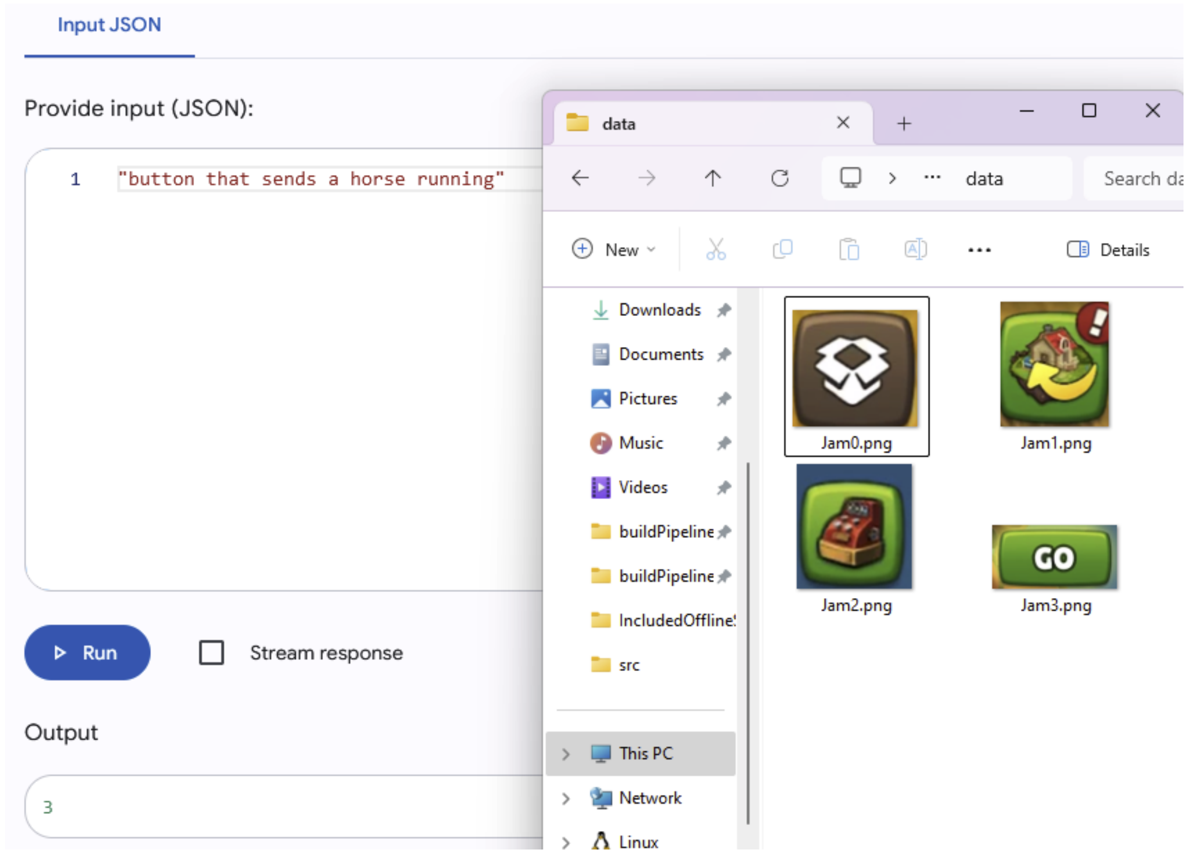
System prompt
Enter a system prompt (optional)

esfsf

User
Add media

hello there2

Model


Input JSON

Provide input (JSON):

```
"button that sends a horse running"
```

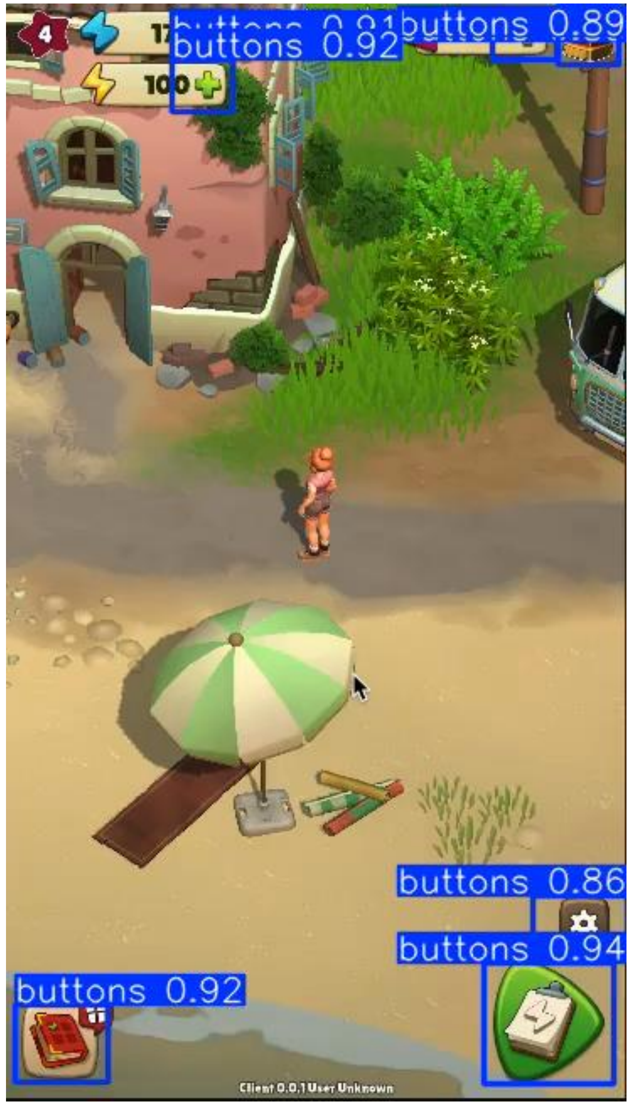
Run
Stream response

Output

```
3
```

BONUS: RUNNING YOLO ON VIDEO

- 32ms processing time per frame on a macbook with M1 pro



05

CONCLUSION

CONCLUSION

- Generating data set through script was a good decision
- Classification is quite powerful (prototype used only one label)
- Would be too expensive to run each time, but the results of the found positions could be stored and used until the test fails again
- Didn’t get it all working in 1 day but was real fun!

END OF PRESENTATION

Thank You

GET IN TOUCH

Finn Kasulke
Senior Quality Assistance Engineer

Johannes Deml
Senior Frontend Developer

Robin Vonsien
Senior Frontend Developer