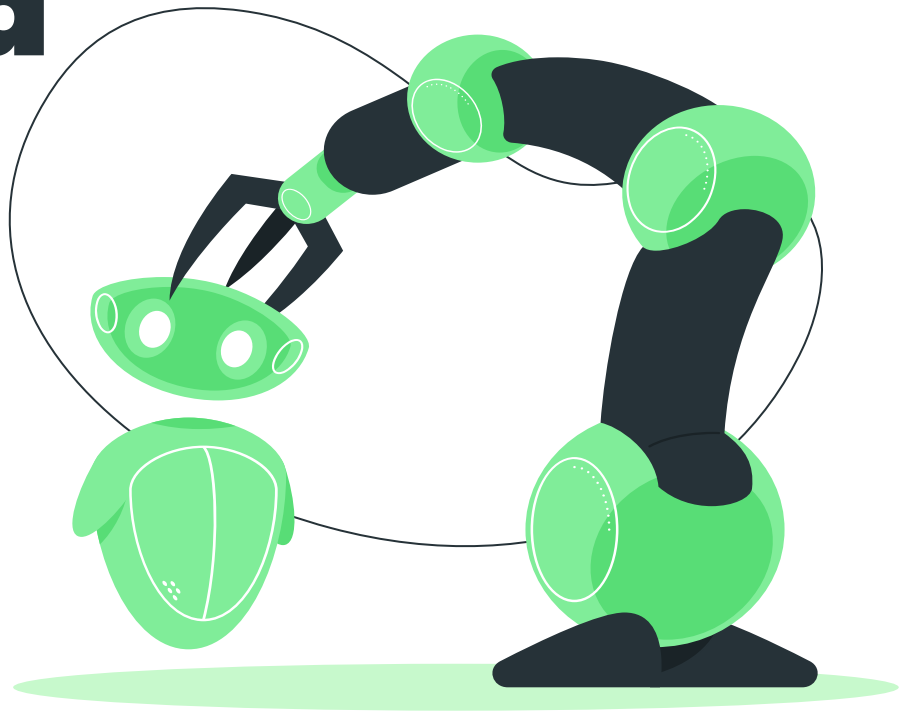
Augmented Reality

Research about augmented reality and demo on Pokémon Go game



- Members:
- Ngo Quang Vinh - KHTN 2019
- Ngo Huu Manh Khanh - KHTN 2019
- Nguyen Minh Huy- KHTN 2019
- Instructor:
- Dr. Mai Tien Dung

Table of Contents

1 Problem Identification

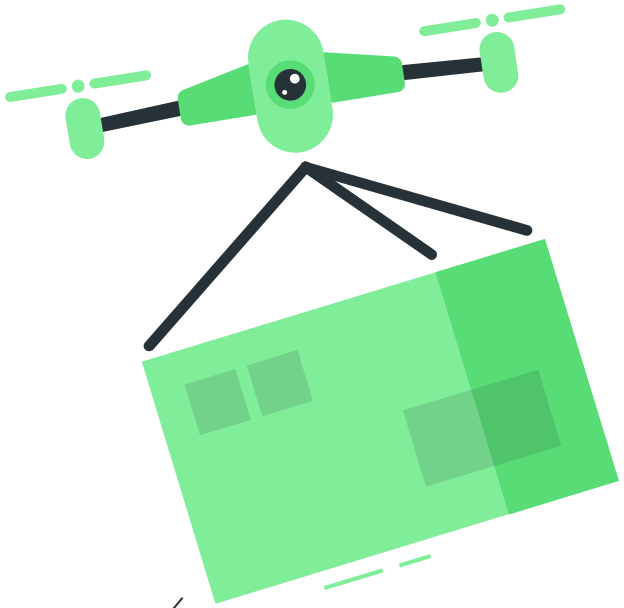
2 CV concept

3 CG concept

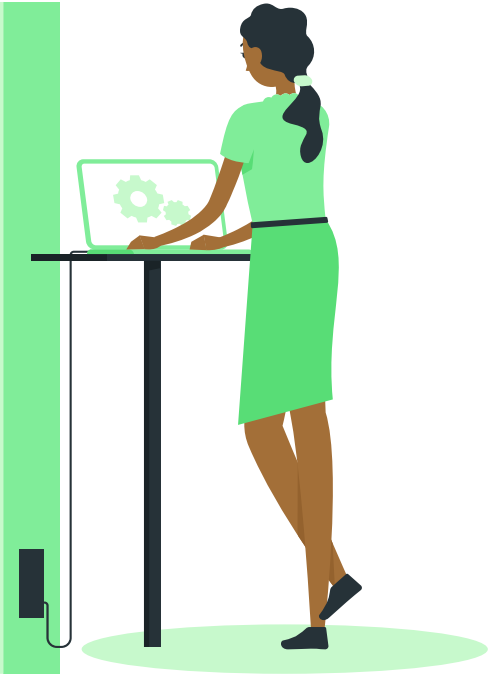
4 Implementation

What is AR ?

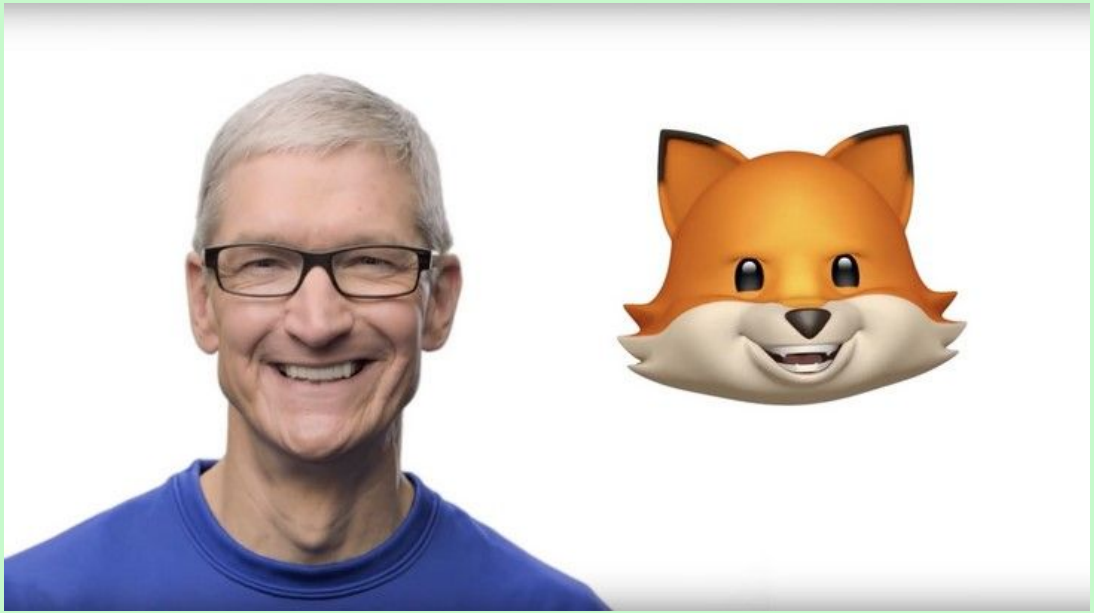
Augmented reality (AR) is an interactive experience of a real-world environment where the objects that reside in the real world are enhanced by computer-generated perceptual information.

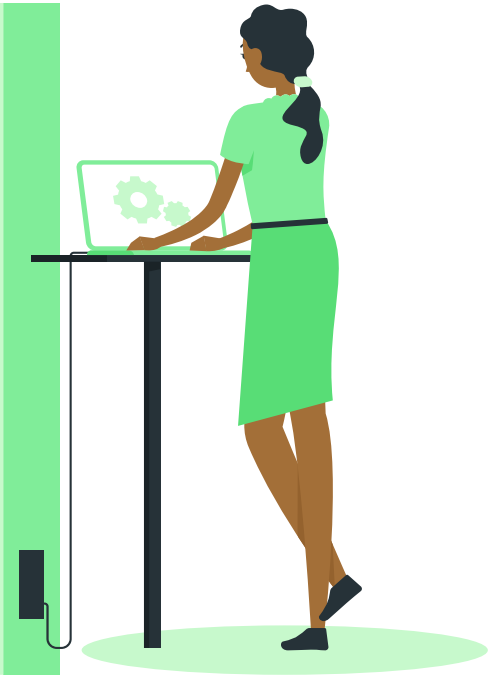


Examples

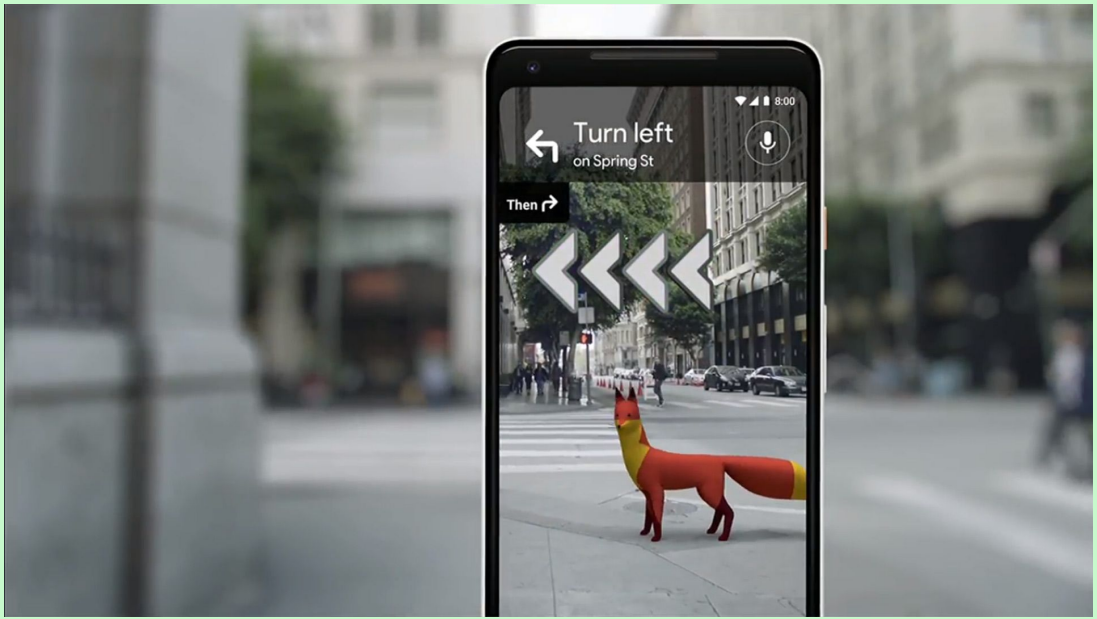


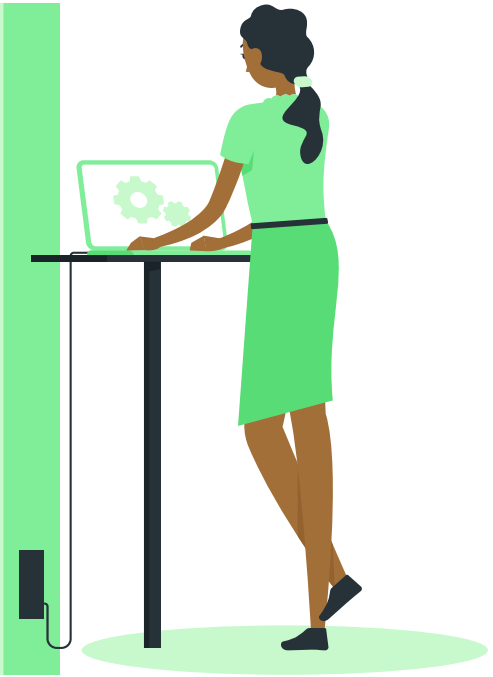
Examples



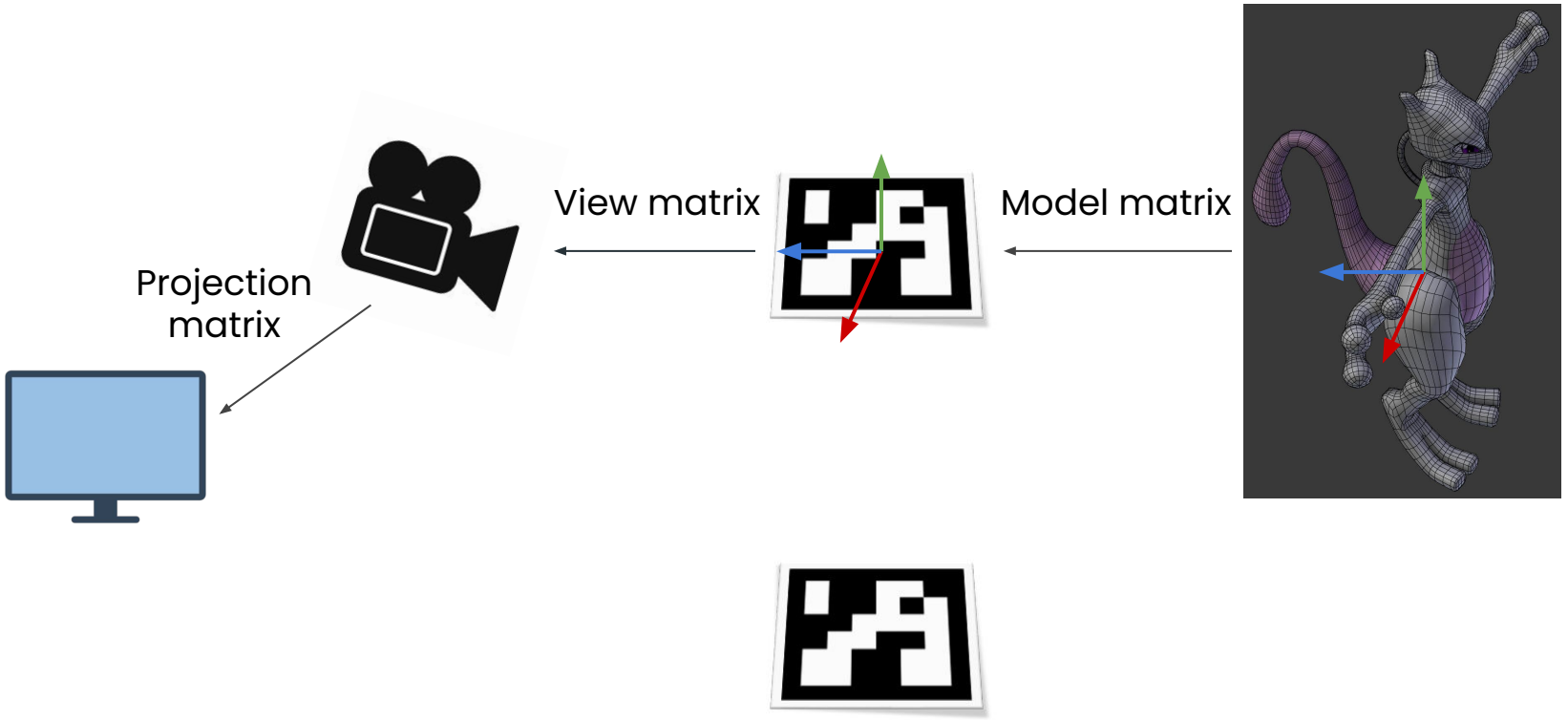


Examples



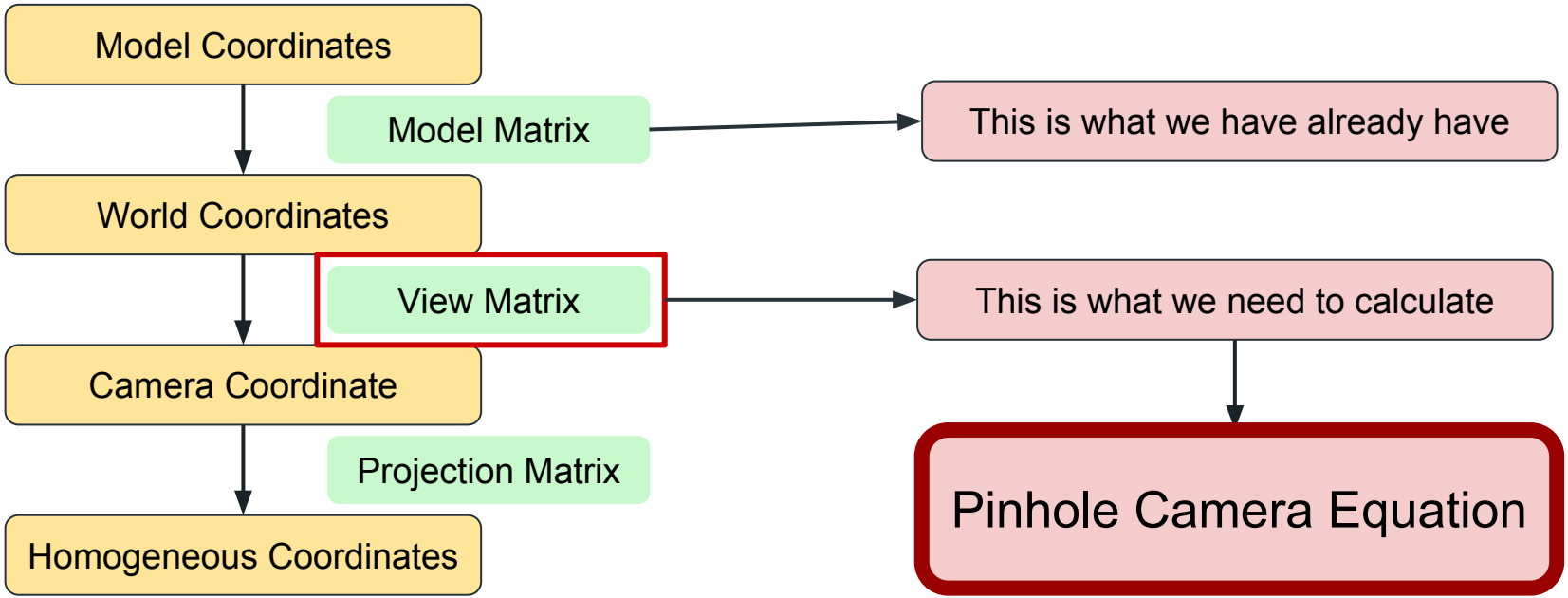


Concepts



Concepts

From previous, we had:



Note: we still not have Projection Matrix but we will Calculate in later Section

Concepts

View Matrix - Pinhole equation

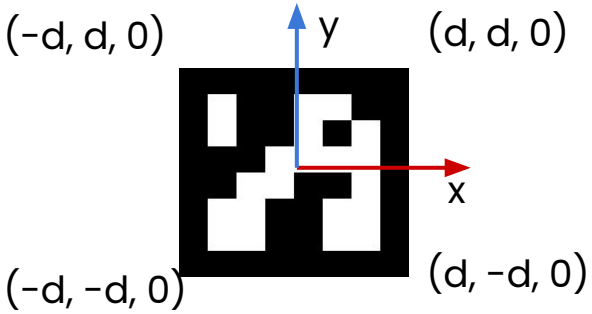
Rotation Matrix

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{12} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Translation Matrix

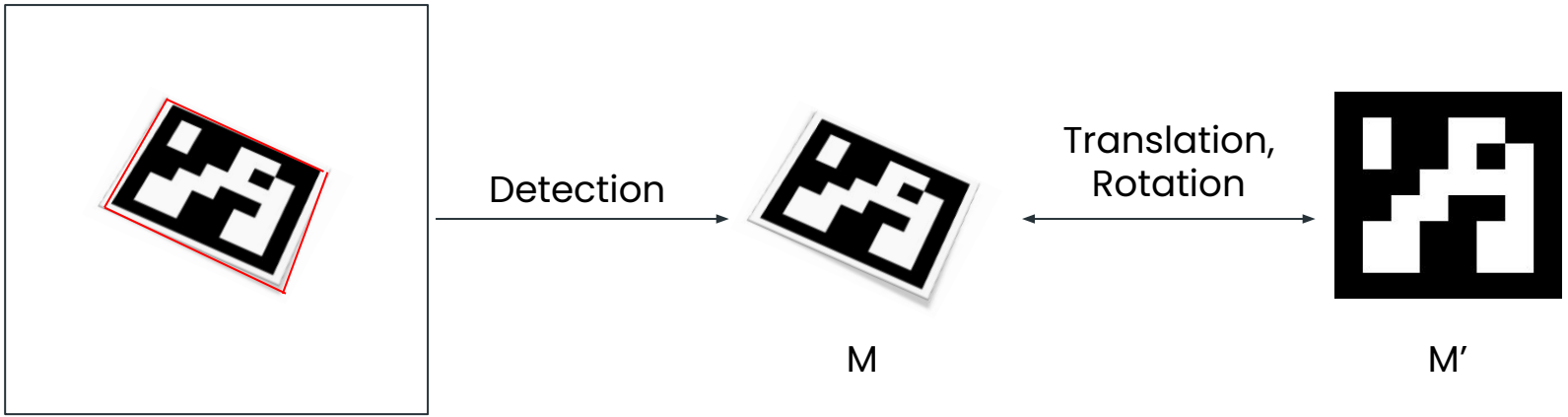
Concepts

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{12} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



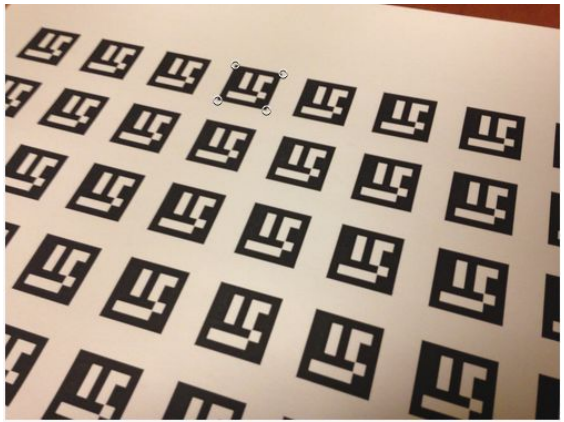
Concepts

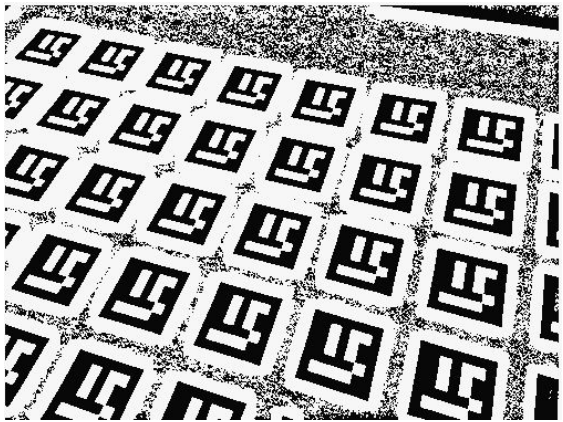
- **View Matrix**

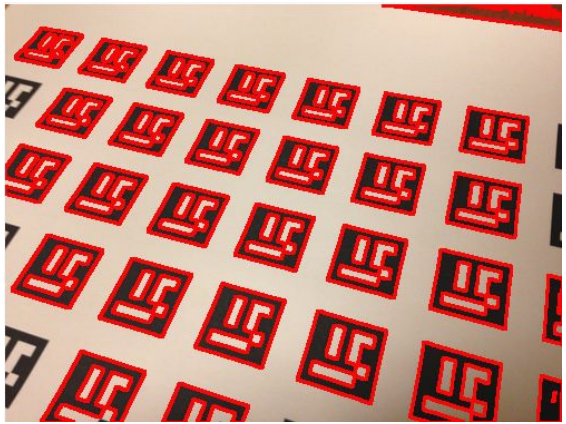


Concepts

- Detect Marker

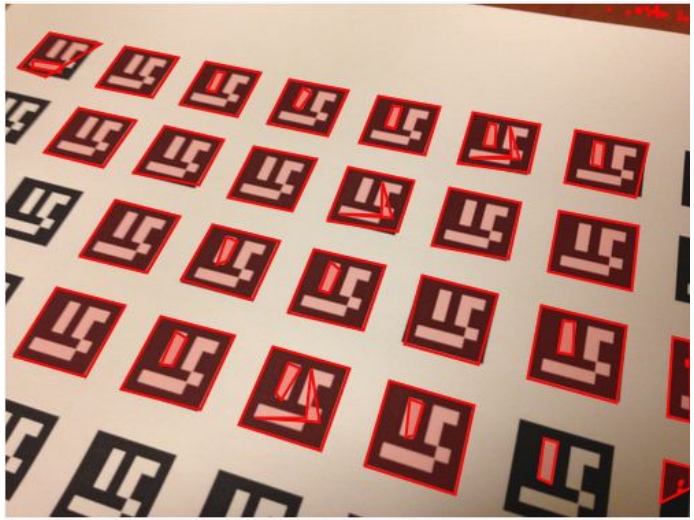


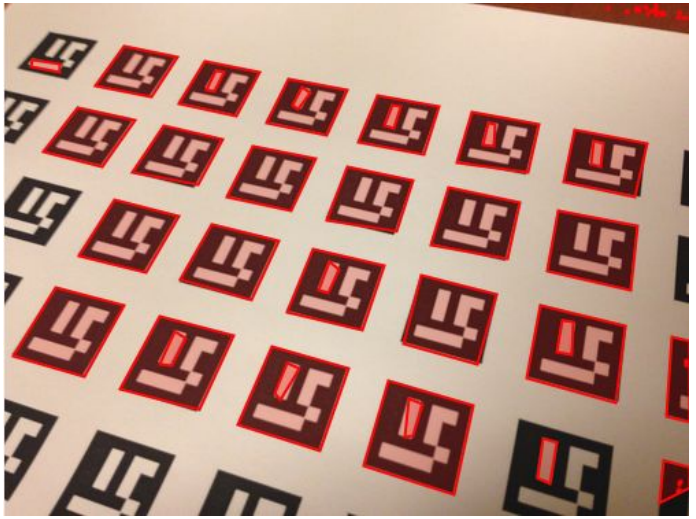




Concepts

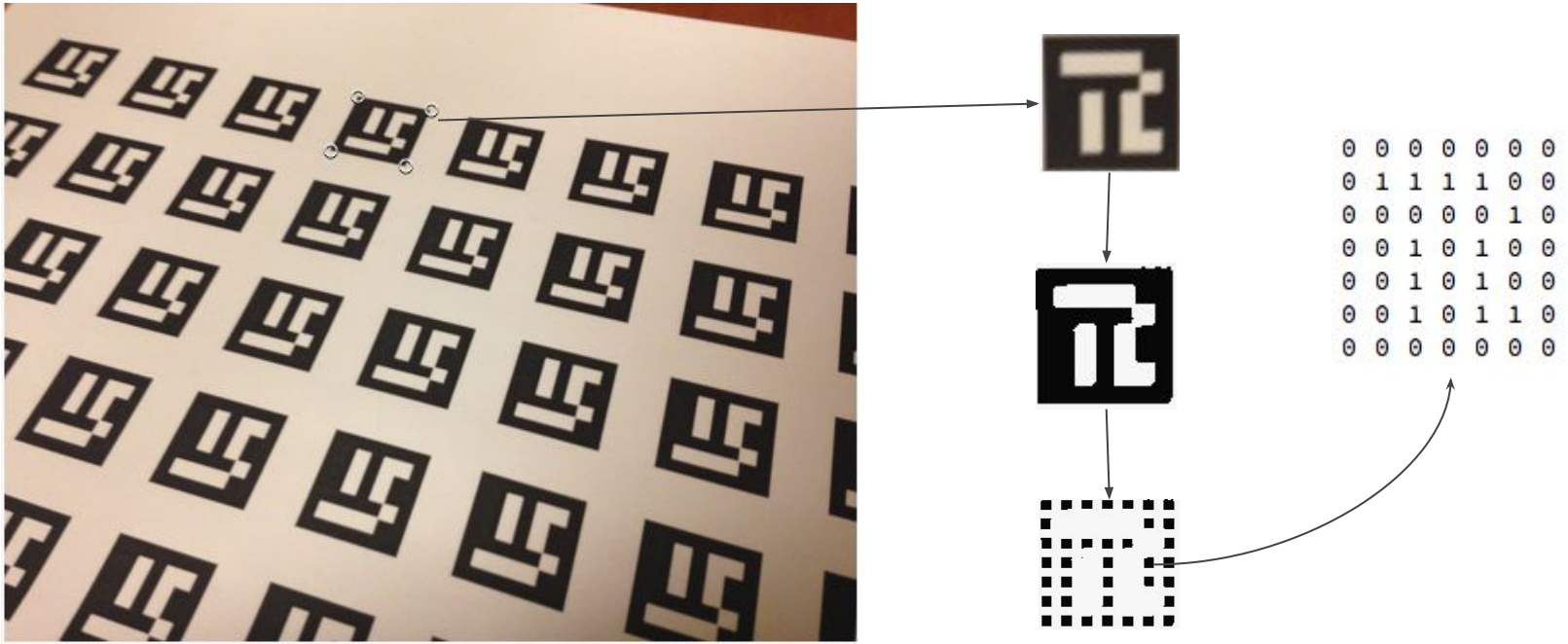
- Detect Marker





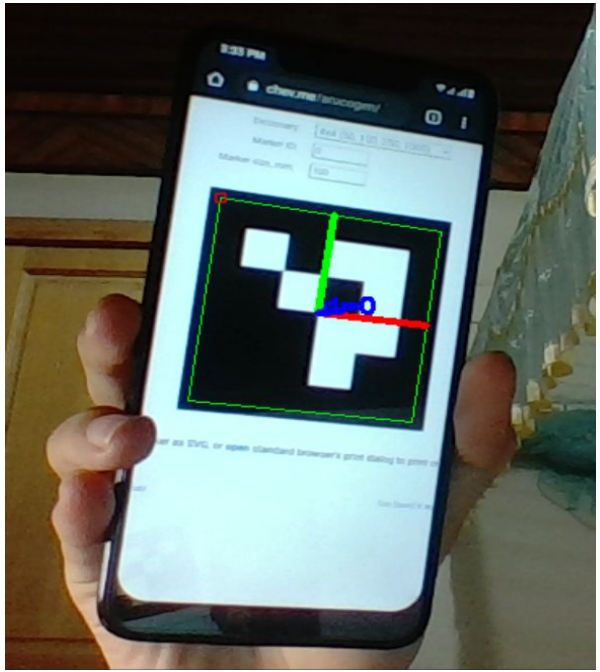
Concepts

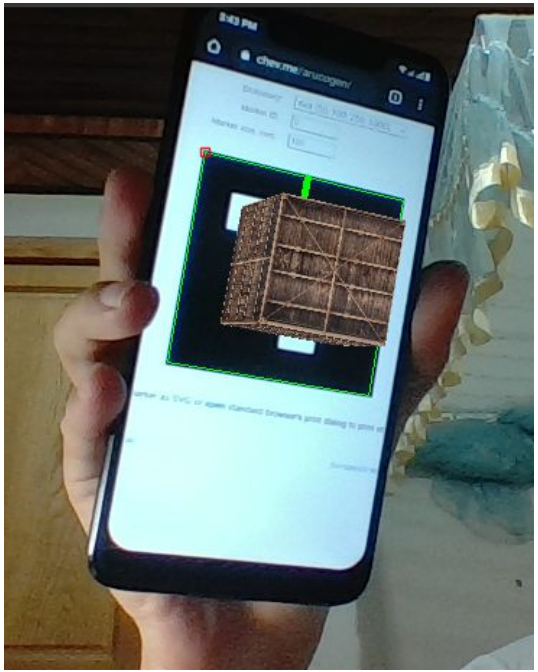
Detect Marker



Concepts

Result



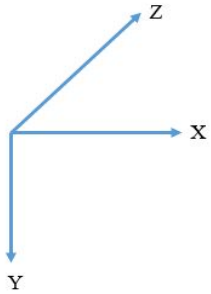


Concepts

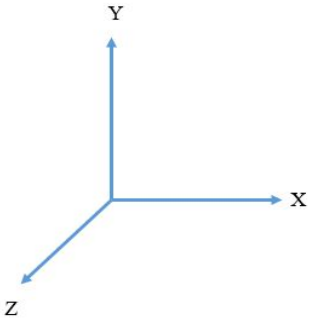
$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{12} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Concepts

Concepts



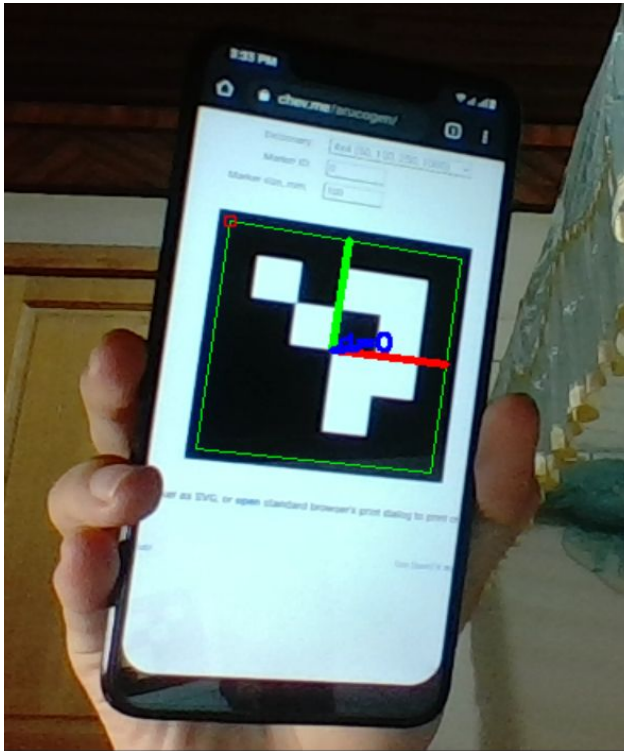
OpenCV camera coordinates



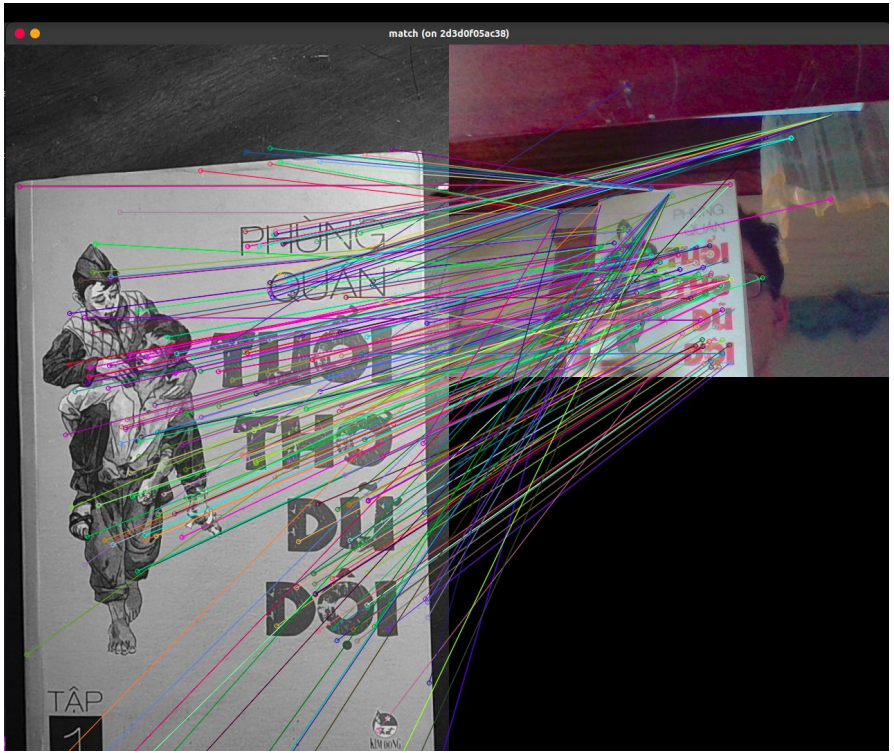
OpenGL camera coordinates

$$\begin{bmatrix} r_{11} & r_{12} & r_{12} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \longrightarrow \begin{bmatrix} r_{11} & r_{12} & r_{12} & t_1 \\ -r_{21} & -r_{22} & -r_{23} & -t_2 \\ -r_{31} & -r_{32} & -r_{33} & -t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Concepts



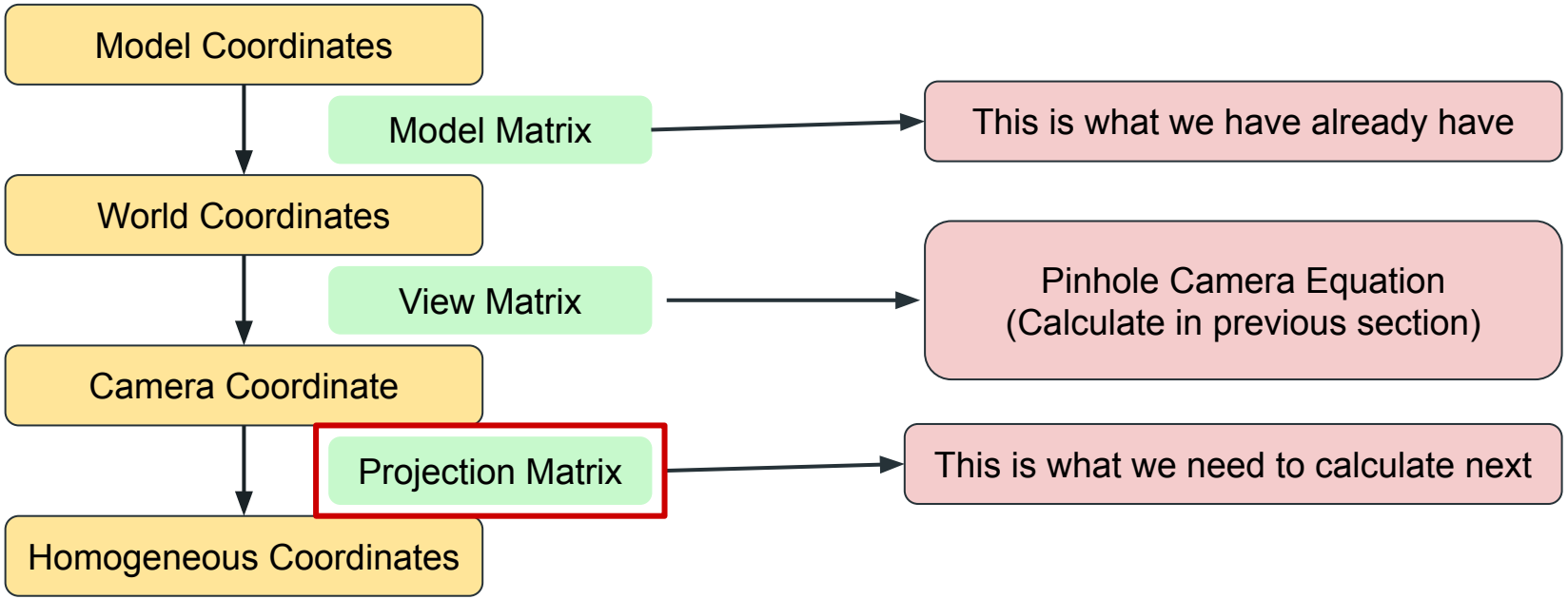
FPS ~ 20-23



FPS ~ 8-10

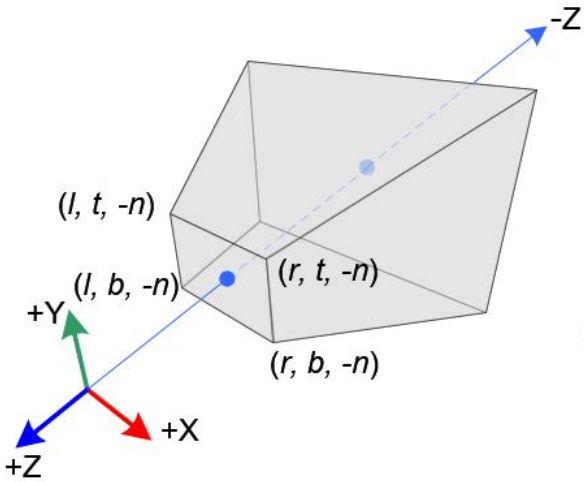
Concepts

From previous, we had:

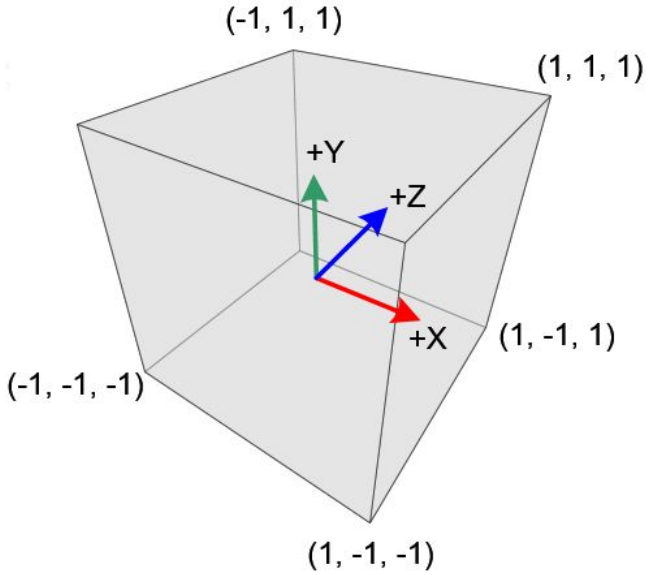


Concepts

Projection Matrix



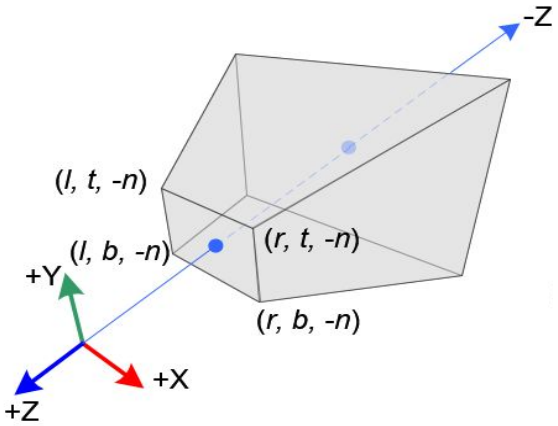
Perspective Frustum



Normalized Device Coordinates (NDC)

http://www.songho.ca/opengl/gl_projectionmatrix.html

Concepts



$$\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

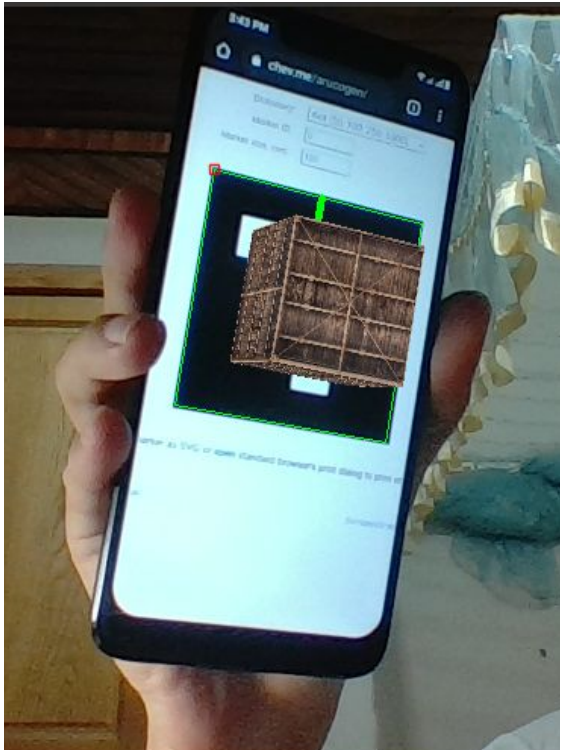
http://www.songho.ca/opengl/gl_projectionmatrix.html

Concepts

$$Projection = NDC.Persp = \begin{bmatrix} \frac{2f_x}{w} & 0 & 1 - \frac{2c_x}{w} & 0 \\ 0 & \frac{2c_y}{h} & \frac{2c_y}{h} - 1 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

http://ksimek.github.io/2013/06/03/calibrated_cameras_in_opengl/

Concepts



Concepts

Concepts

Problem identification

A

Shader

Appropriate levels of light, darkness, and color

B

Force & Direction

Adjust balls ‘s force and direction.

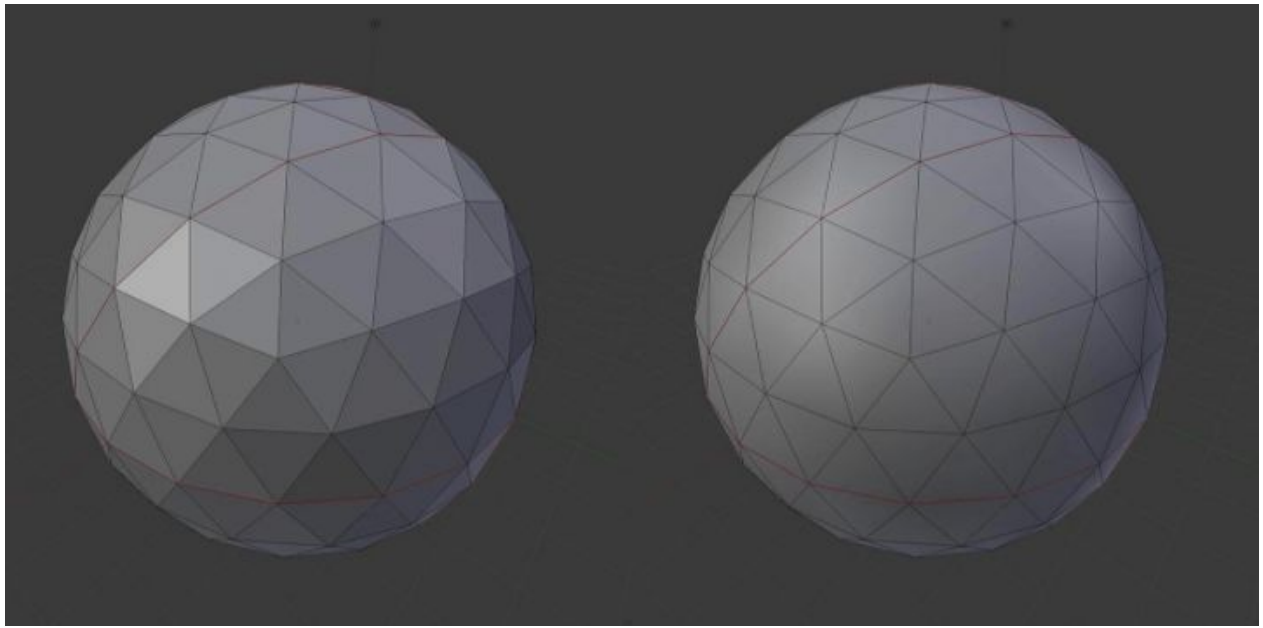
C

Collision detection

How do we know if the ball touches an object and bounces back

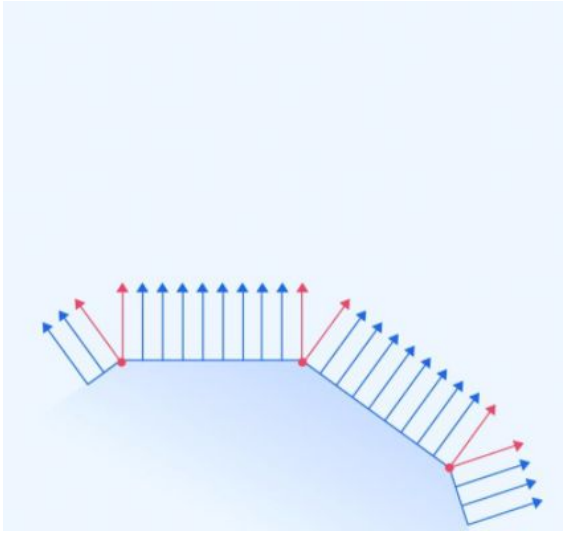
Concepts

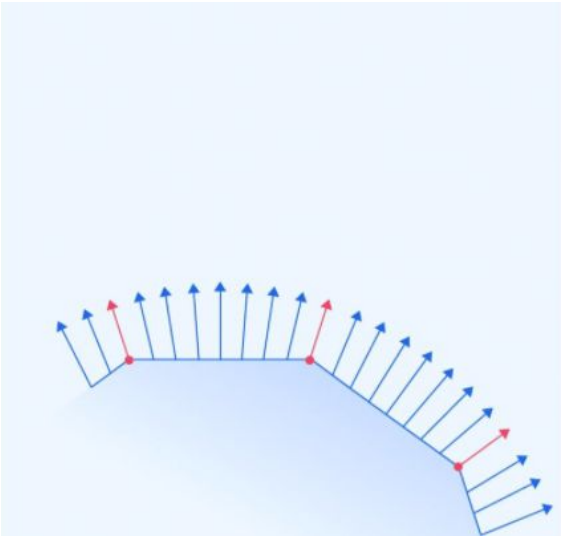
A Shader



Concepts

A Shader

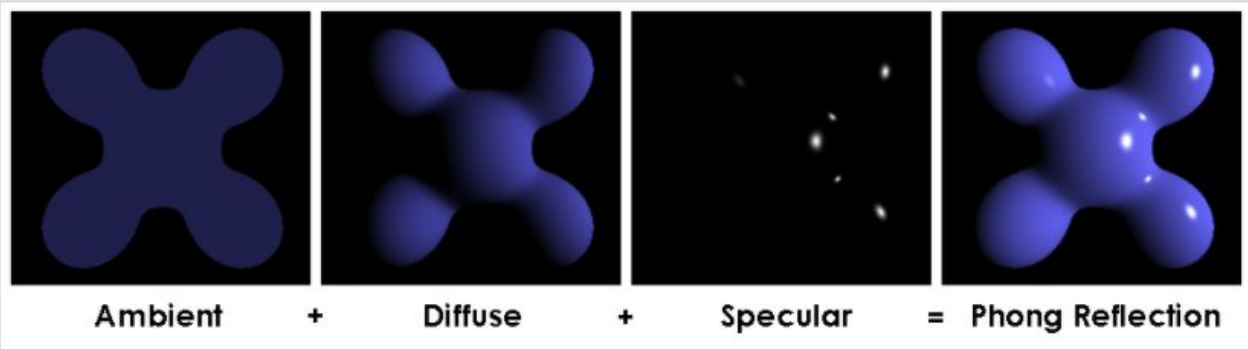




Concepts

A Shader

- Phong model



Visual illustration of the Phong equation: here the light is white, the ambient and diffuse colors are both blue, and the specular color is white, reflecting a small part of the light hitting the surface, but only in very narrow highlights. The intensity of the diffuse component varies with the direction of the surface, and the ambient component is uniform (independent of direction).

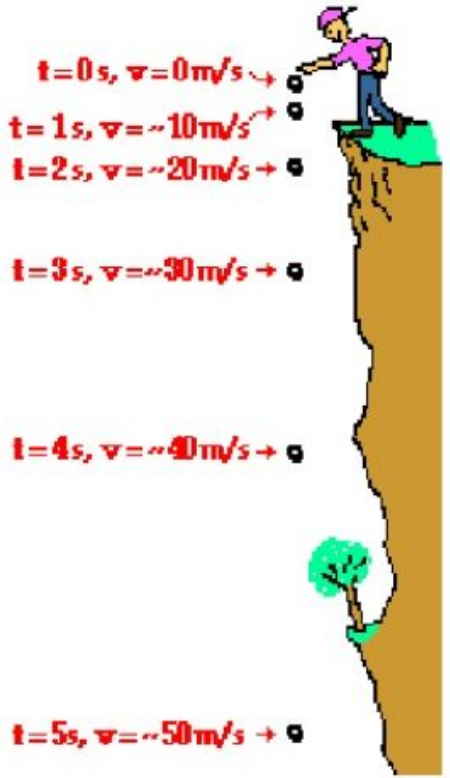
Concepts

B Force & Direction

- Gravity (y axis)

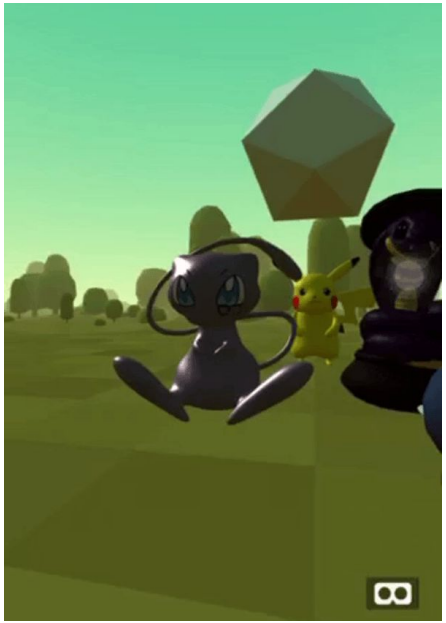
$y = y + v$
$v = v_0 + g * t$

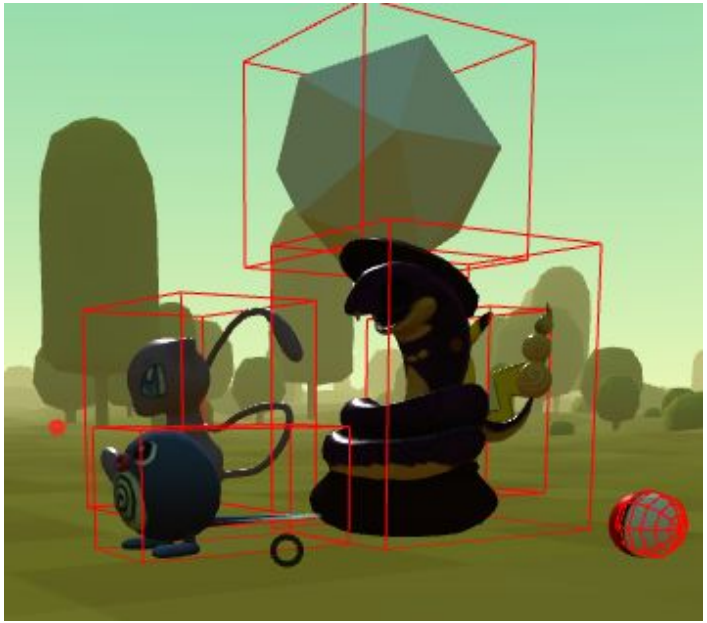
- Where:
- y is the coordinate along the vertical axis
- v is the displacement of y per frame
- v_0 is initial velocity
- g is gravitational acceleration in this axis or acceleration (a) in general



Concepts

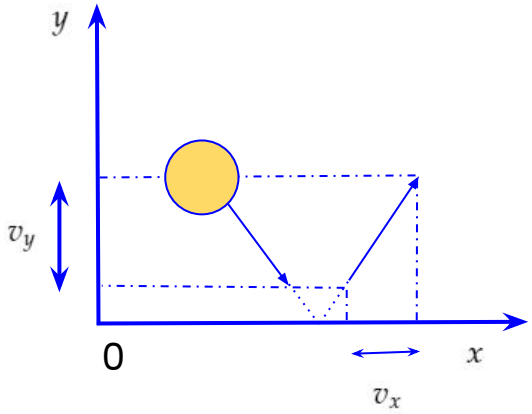
c Collision detection

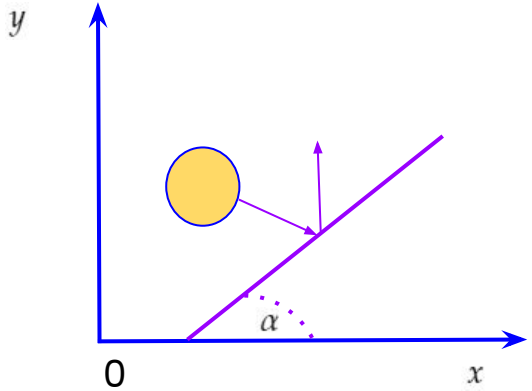




Concepts

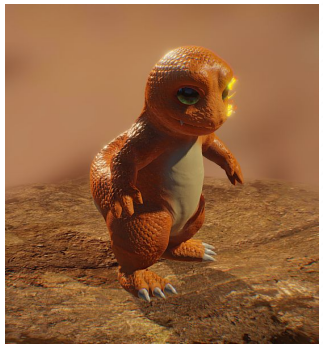
c Collision detection

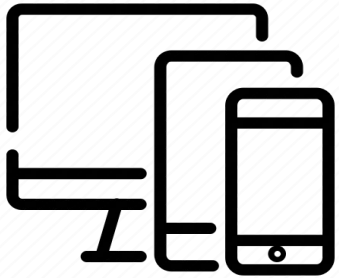


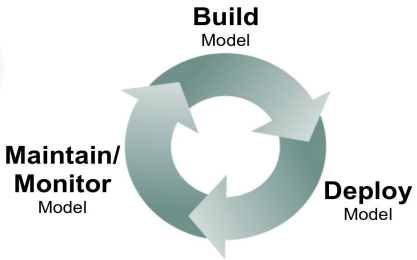


Implementation

- Based on baseline and physics systems we propose in previous, we come up to the demo implementation.
- Target:
  - Lightweight applications
  - Multiplatform
  - Easy to deploy and maintain







Implementation

- As all targets and all problems we have researched, We proposed to deploy our demos into Web environments.

- Deployment

GitHub Pages

- Libraries and Frameworks

HTML

WebGL

three.js

A-FRAME

Why this?

Implementation

- As easy to debug and demonstrate the physics system of the game application, we provide 4 versions of the game based on:

- Web VR versions

- Web AR versions

- Web VR versions with physics

- Web AR versions with physics

Implementation

How to play on PC

- Use keyboard keys `A W S D` and `mouse` for moving around and changing view in PC.
- Pressing `space` for shooting in PC. Velocity of the bullets are based on how long you press the button.
- If you have VR devices, you can click the toggle button in right bottom. It would allow you to use VR devices.

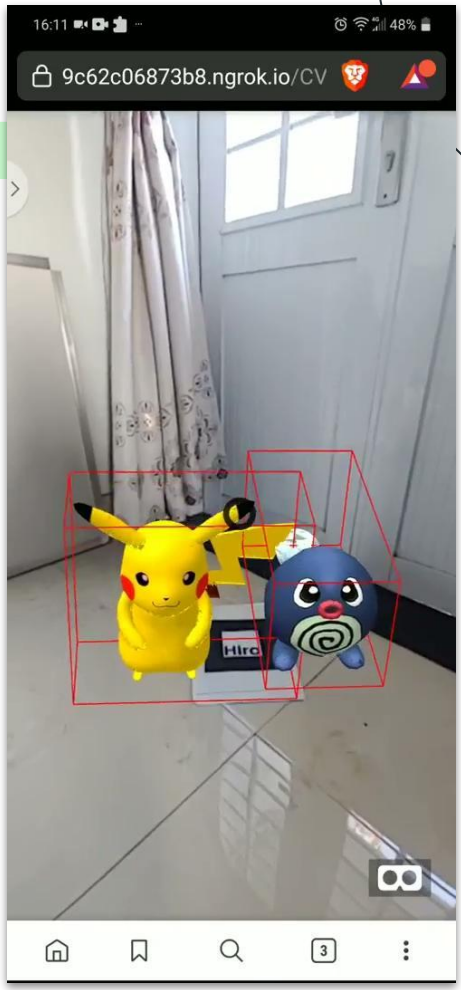
Try ➠

Implementation

How to play on Mobile device

- Use gestures `=>` (swipe right) and `<=` (swipe left) or moving your camera around for changing view in Mobile.
- Use gesture `swipe up` for shooting in Mobile. Velocity of the bullets are based on how fast you swipe
- If you would like to try a VR edition, you should take this photo as a marker:

Try ➠



Now, what we have?

Implementation - Now

Github pages environment for documentation and demo

- In our repo, we provide a plug-and-play source code which you only need to pull a docker image and run makefile to compile executed code.

Computer Graphics and Computer Vision

Description:

This repository is used for storing sourcecode related to final project of Computer Graphics and Computer Vision

Table of content:

1. Docker image for Environment
2. Sourcecode for the project
    1. Baseline sourcecode
    2. Implemented sourcecode
        - About this game

Implementation - Now

Github pages environment for documentation and demo

- In our repo, we provide a plug-and-play source code which you only need to pull a docker image and run makefile to compile executed code.

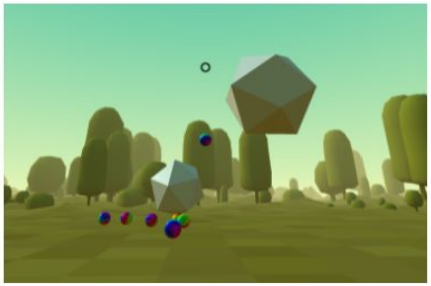
Game application demos

- We also provide our game application demo which is hosted directly in github pages so everyone can try our game

◎ Demo

- 3D prototype game
- 3D prototype game with physics
- Pokemon Go game
- Pokemon Go game with physics

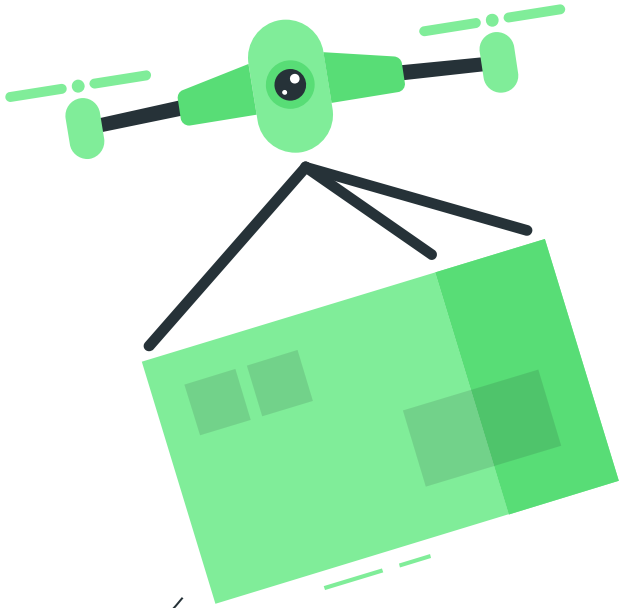
◎ How to play



- Use keyboard keys `A W S D` and `mouse` for moving around and chaging view in PC.
- Use gestures =>(swipe right) and <=(swipe left) for changing view in Mobile.
- Pressing `space` for shooting in PC. Velocity of the bullets are based on how long you press the button.
- Use gesture `swipe up` for shooting in Mobile. Velocity of the bullets are based on how fast you swipe.

What is next?

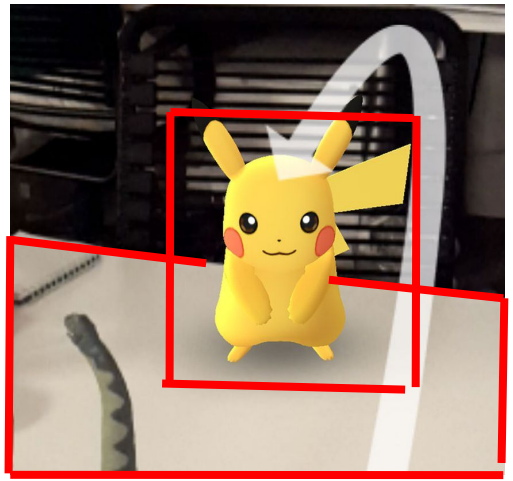
In the future, we will continue to maintain and upgrade this application



Implementation - Future

Further than AR marker

- Our AR game only work with marker, which is a nuisance. In future we will try to detect the landscape for more convenience.



Implementation - Future

Further than AR marker

- Our AR game only work with marker, which is a nuisance. In future we will try to detect the landscape for more convenience.

More game content/ animations

- We will update our animation in the game to make it more realistic: shading algorithm, physics system, ...

Implementation - Future

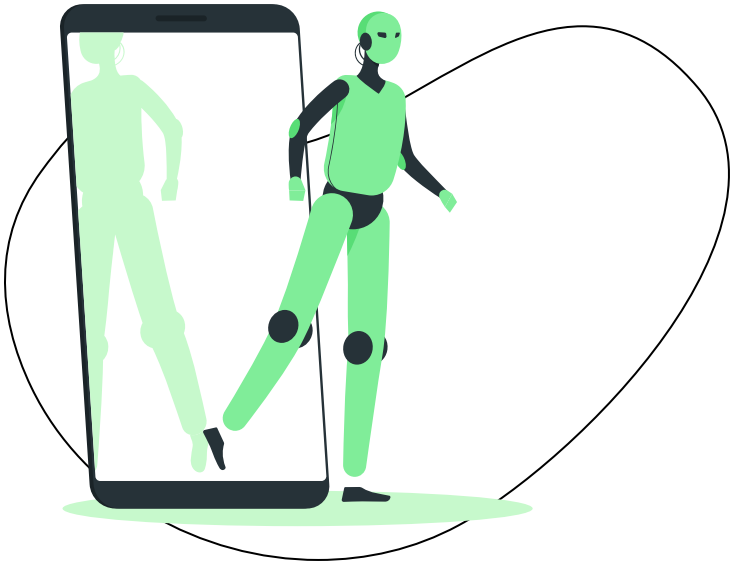
Further than AR marker

- Our AR game only work with marker, which is a nuisance. In future we will try to detect the landscape for more convenience.

More game content/ animations

- We will update our animation in the game to make it more realistic: shading algorithm, physics system, ...
- Now our game is really simple, but we will upgrade with more storylines, characters, levels, also connection between players

Thanks for your listening!



Do you have any questions?

If you love and would like to keep track of our project, star us at:

https://github.com/vinhqngo5/Computer-Vision-And-Computer-Graphics

Scan me