Using Face Detection as a Game Input Device

Andrew Brusso

Michigan Technological University

April 25, 2020

What I Wanted to Accomplish

1. Create a game controlled using a webcam.
2. Use OpenCV for image processing and facial detection.
3. Understand basic aspects of the Unity3D game engine.
4. Familiarize myself with Python and *C#*.

The Challenges

- No OpenCV, Unity3D, *C#*, or game development experience. Limited Python experience.
- Webcams are noisy input devices, and don't generally have a high framerate.
- Unity doesn't have native ways to handle custom input devices or running python code.
- Facial Detection is a hard problem to solve from scratch (good solutions require deep neural nets).

How I handled: Using a New Toolkit

- For OpenCV and Unity: followed lots of tutorials to learn the basics.
- The Unity tutorials helped a lot with *C#*, but its syntax is also similar to C++.

Unity tutorial series by Brackeys

How I handled: Reining in Latency, Reducing Noise

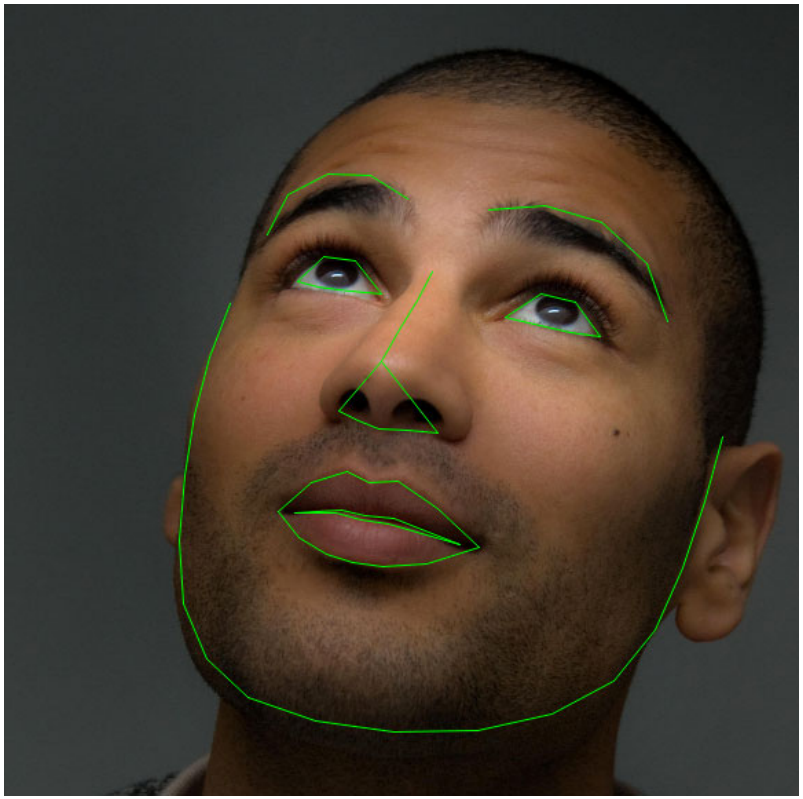
- I/O operations are slow:
  - Created 3 separate threads for retrieving frames from camera, sending them to unity, and the actual processing pipeline.
- Cameras are noisy:
  - Used an averaging filter to help smooth inputs out.
  - Used Unity function `RotateTowards`, which does a smoothed out rotation overtime to fill in frames where data is missing.

How I handled: Unity not Having Native/Standard Options

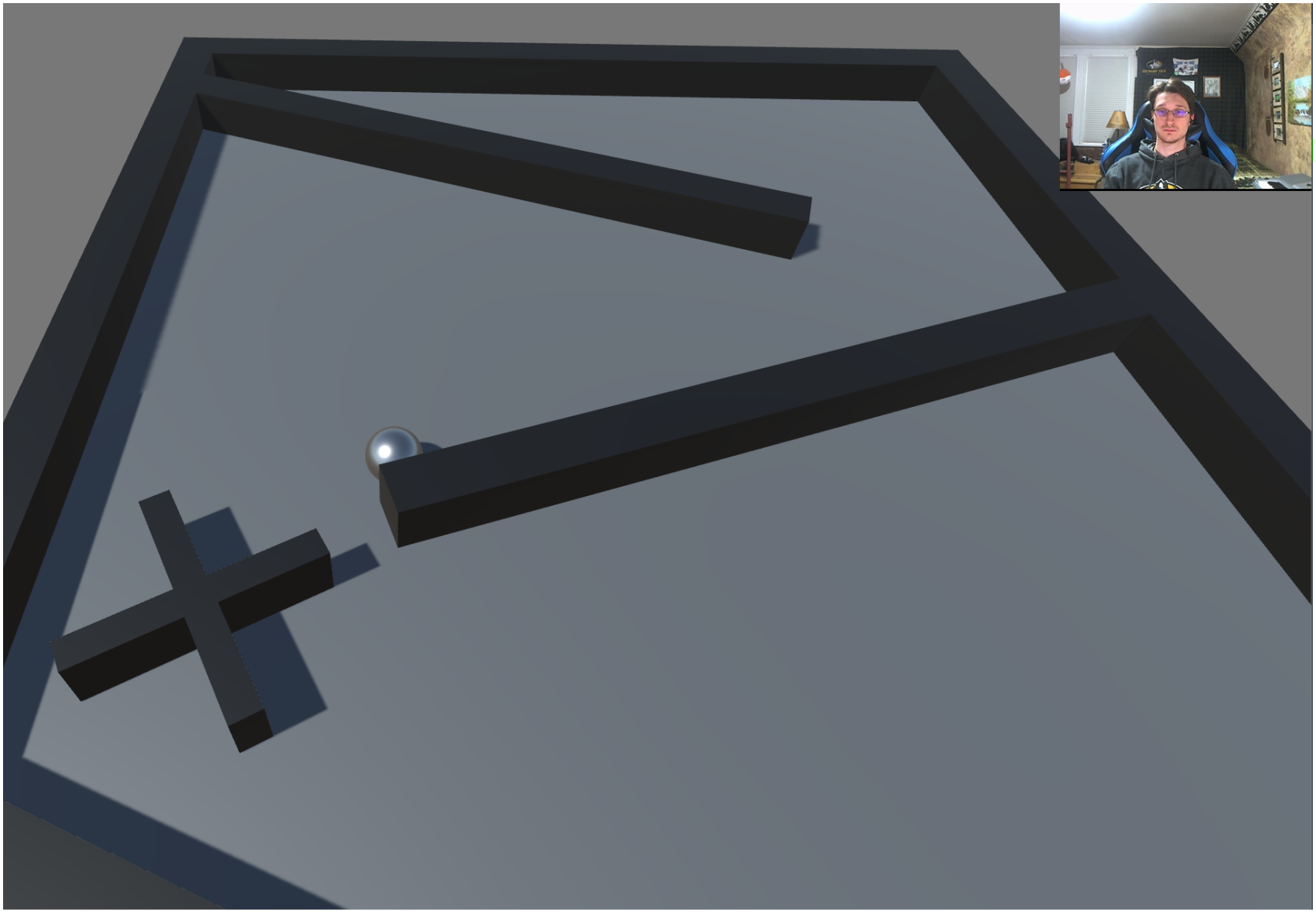
- To run Python, tried:
  - IronPython – An implementation in *C#* of the Python Interpreter – issues with library support.
  - Creating a new process running Python in *C#*, this worked well.
- To create a custom input device:
  - Used a networking library called ZeroMQ (NetMQ in *C#*).
  - Communication over loopback address from a camera server to client inside of Unity.

How I handled: Doing Face Detection

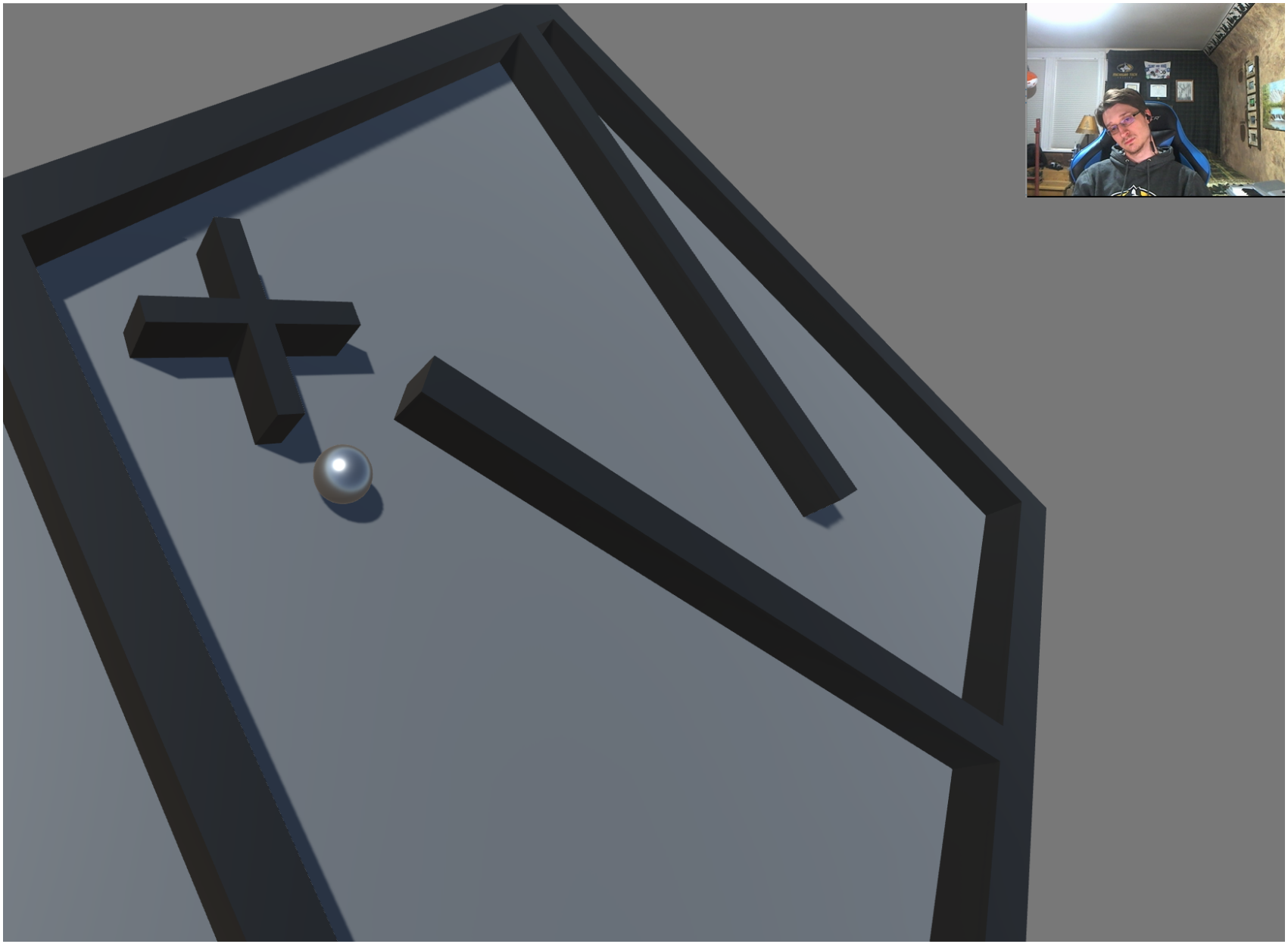
- Used library Dlib, which has built in Deep Learning classifiers optimized for facial detection.
- Dlib has a shape prediction model, which can identify facial landmarks such as eyes, nose, face outline, and mouth.
- Focused on face orientation, by eye positions relative to each other.



The Results



The Results



Possible Future work

- Detect eye closing. Use this as a mechanism for freezing the player. Player would have to keep track of timings while eyes closed.
- Improve some of the game aspects, such as more level designs and game mechanic ideas, since focus has been on face detection and integration so far.

Retrospective

- Latency and accuracy are challenging to balance in a way that makes the controls feel responsive.
- Controlling using webcam face detection is cool, doesn't require expensive hardware, and I could see it being a useful input device for people with motor impairments.