THE OHIO STATE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

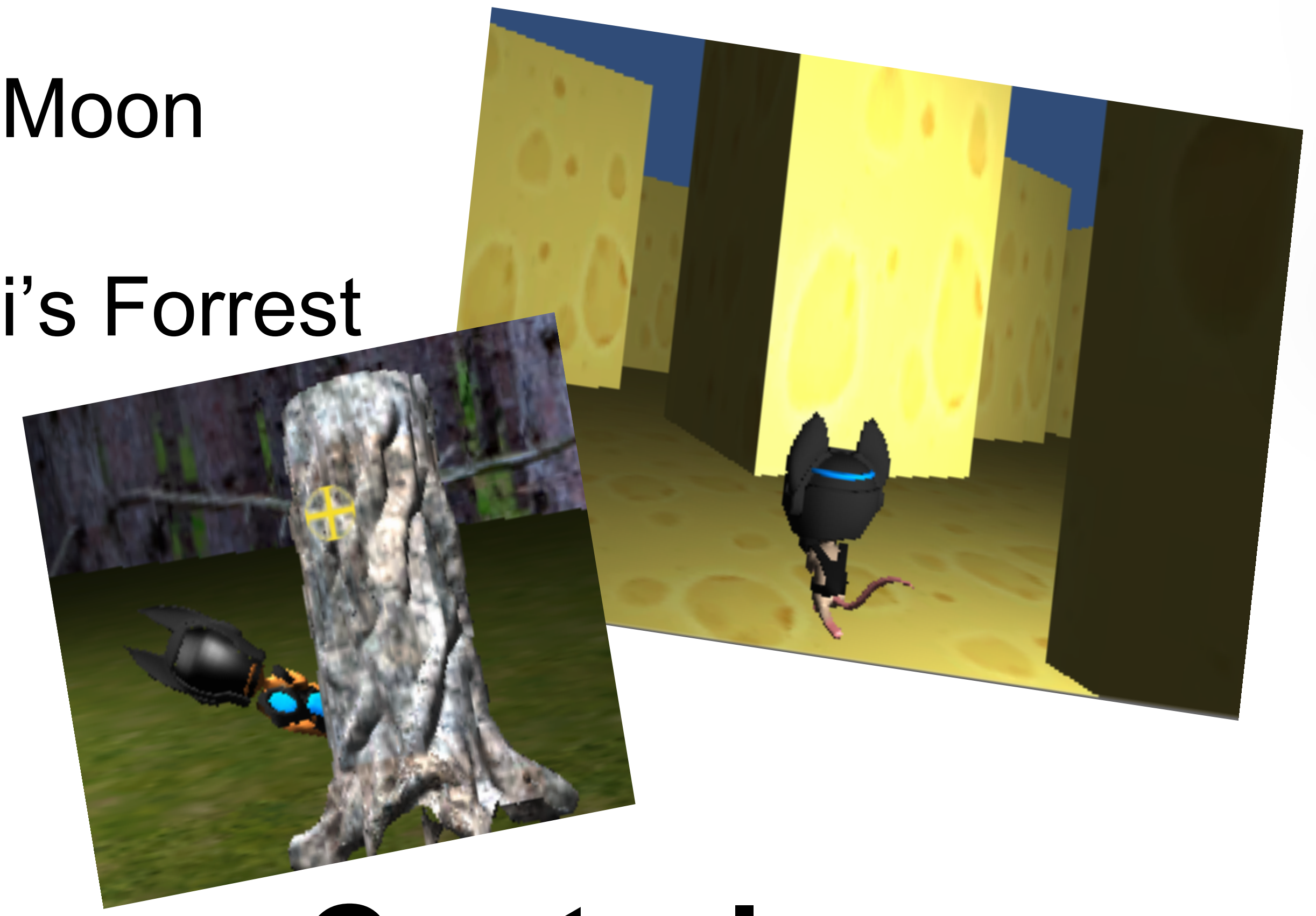
MISSION: IMPAWSIBLE

TEAM ROCKET
game development

Single Player Mini-Games

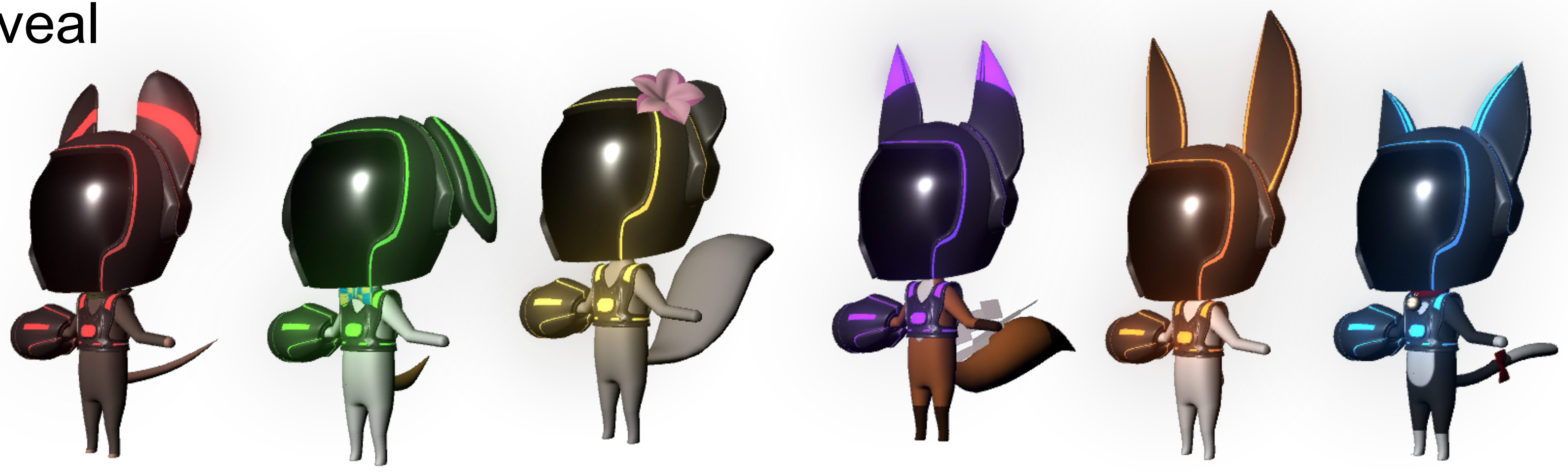
Defeat the hoard of mini-Rosso's as well as many other single player mini-games to reveal a masked captor's identity!

- ★ Rosso's Moon Landing
- ★ Murasaki's Forrest
- ★ More..!



Characters

Play as one of six characters:

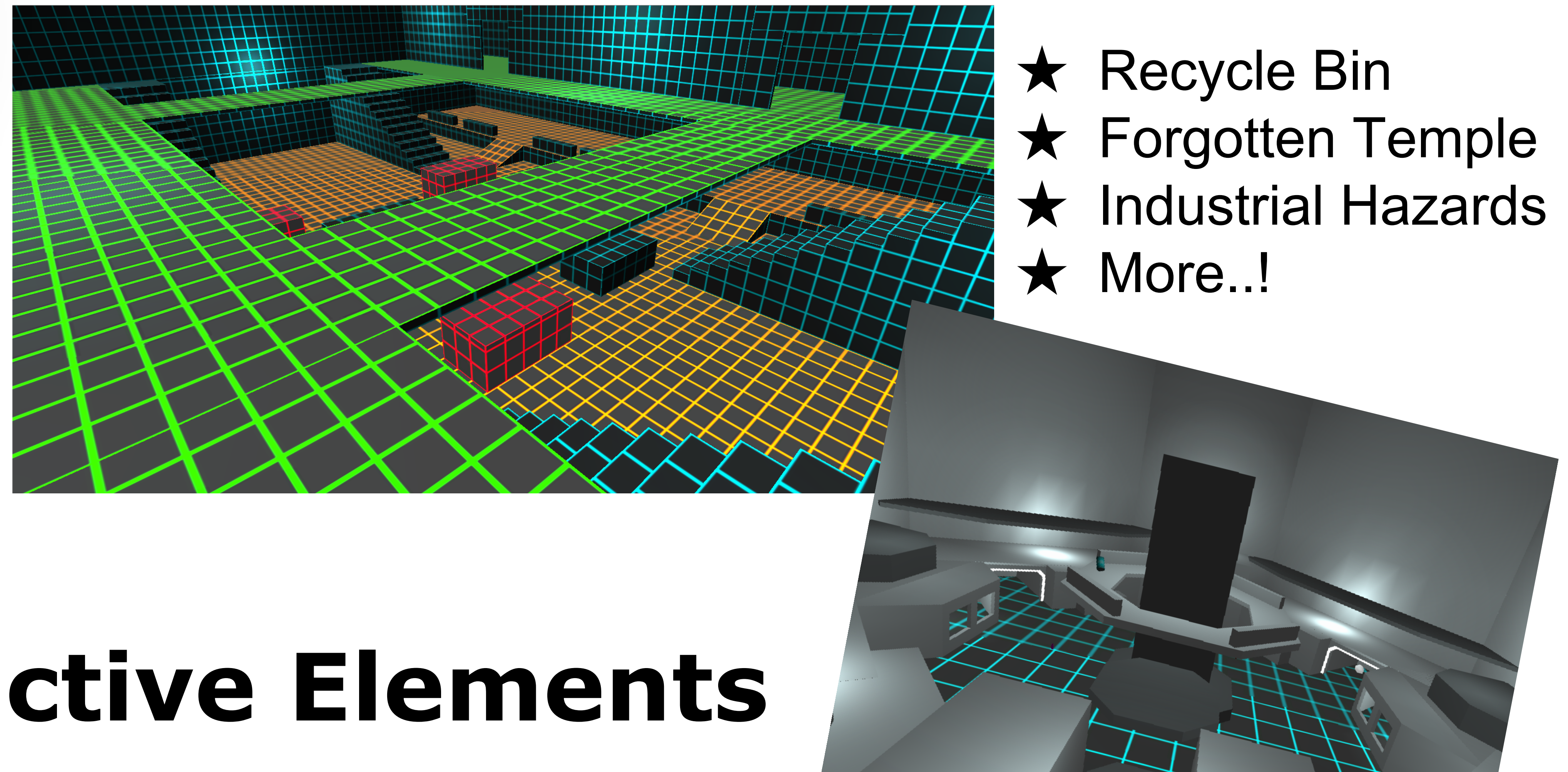


Rosso (Mouse)	Vert (Dog)	Amber (Squirrel)	Murasak i (Fox)	Tangelo (Rabbit)	Lapis (Cat)

Multi-Player Maps

Play up to six friends on one of the multiplayer maps! Face off in timed or stock battles.

- ★ Recycle Bin
- ★ Forgotten Temple
- ★ Industrial Hazards
- ★ More..!

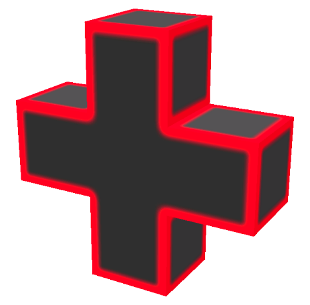


Controls

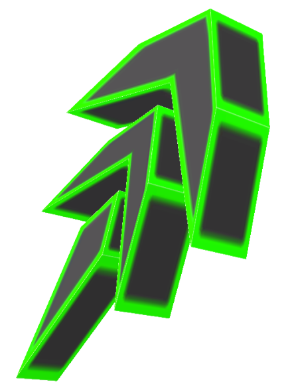
	Keyboard	Playstation Controller
Movement	WASD	Left Joystick
Jump	Spacebar	X button
Vision	Mouse	Right Joystick
Shoot	Left Mouse	R1
In-Game Stats	F3	Select
In-Game Settings	F2	Start
Chat	Enter + Keys	-

For Kinect controls see technical poster.

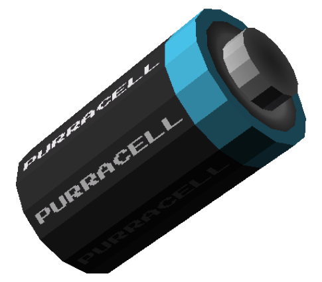
Items



Health Pack
Increases Player's Health



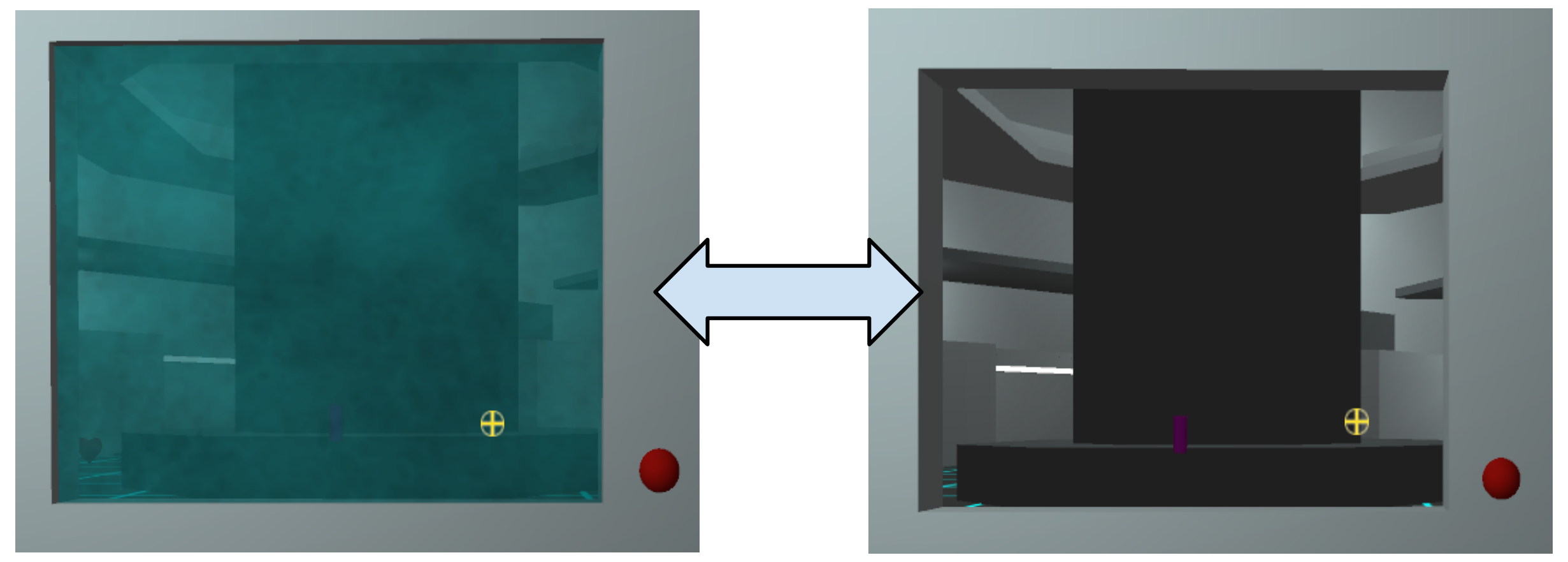
Cat Nip
Increases Player's Speed



Purracell Battery
Recharges Player's Shooting Stamina

Invisibone
Makes You Invisible to Other Players

Interactive Elements



Some maps include buttons that create and destroy barriers.

Developers:

Kimberly Boydstun
Ian Freshwater
Michael Filliater
David Hazlett

Contributors:

Maddy Baringtang
Trevor Richards
Sam Waldron

THE OHIO STATE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

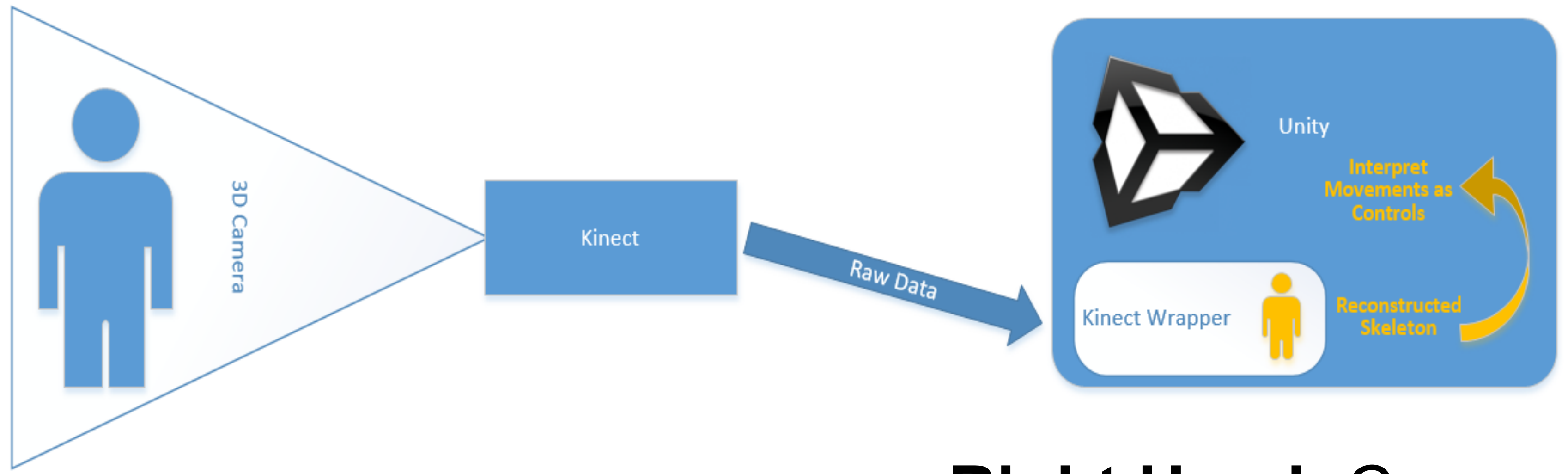
MISSION: IMPAWSIBLE

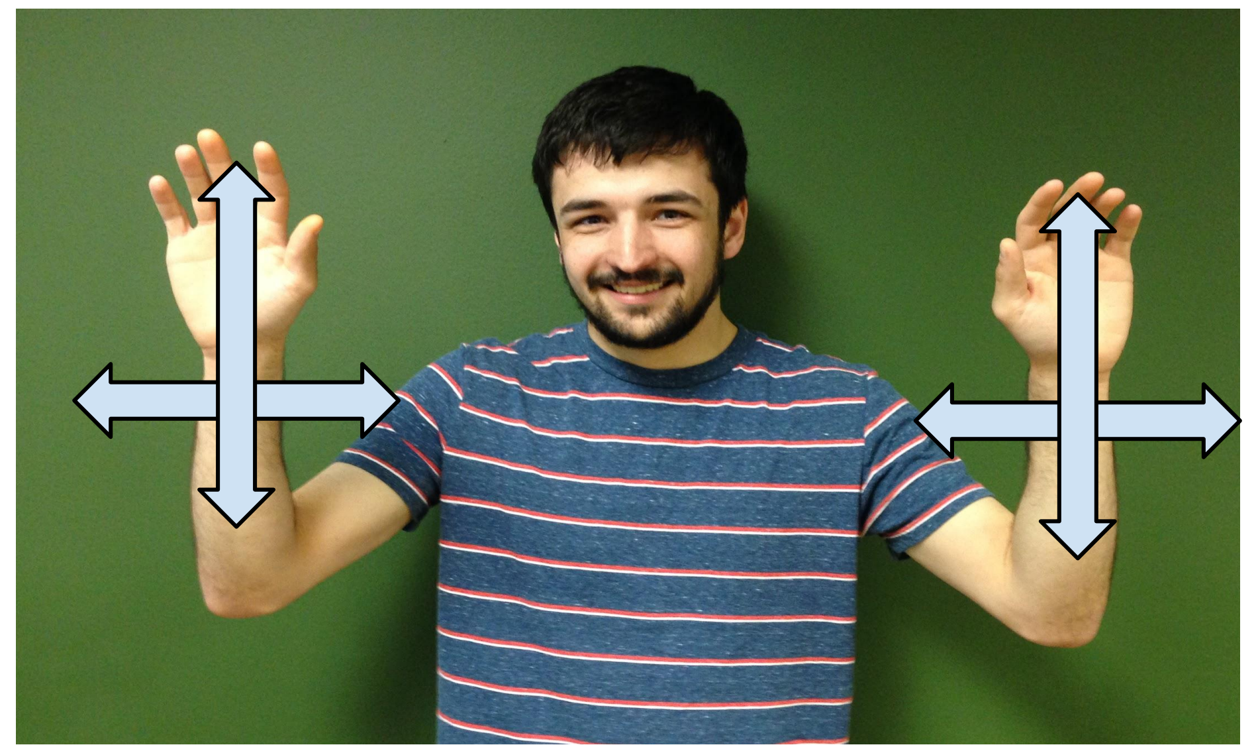
TEAM ROCKET game development

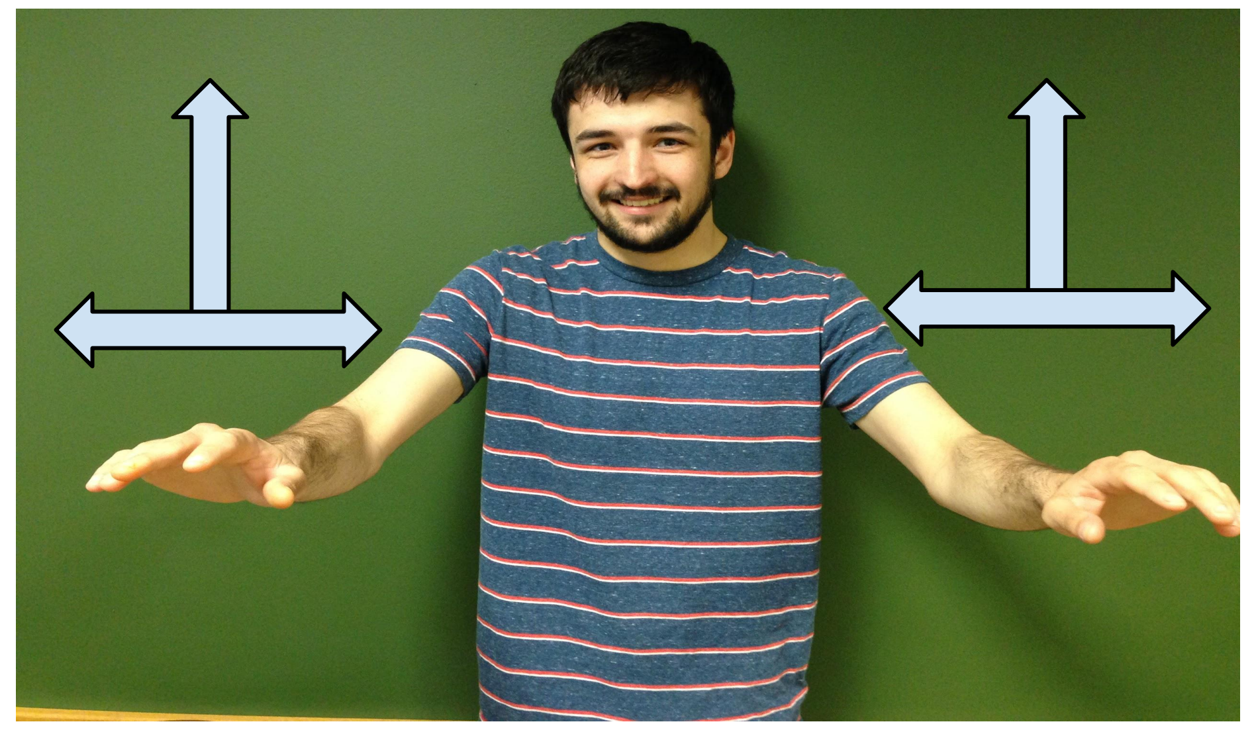
photon UNITY NETWORKING

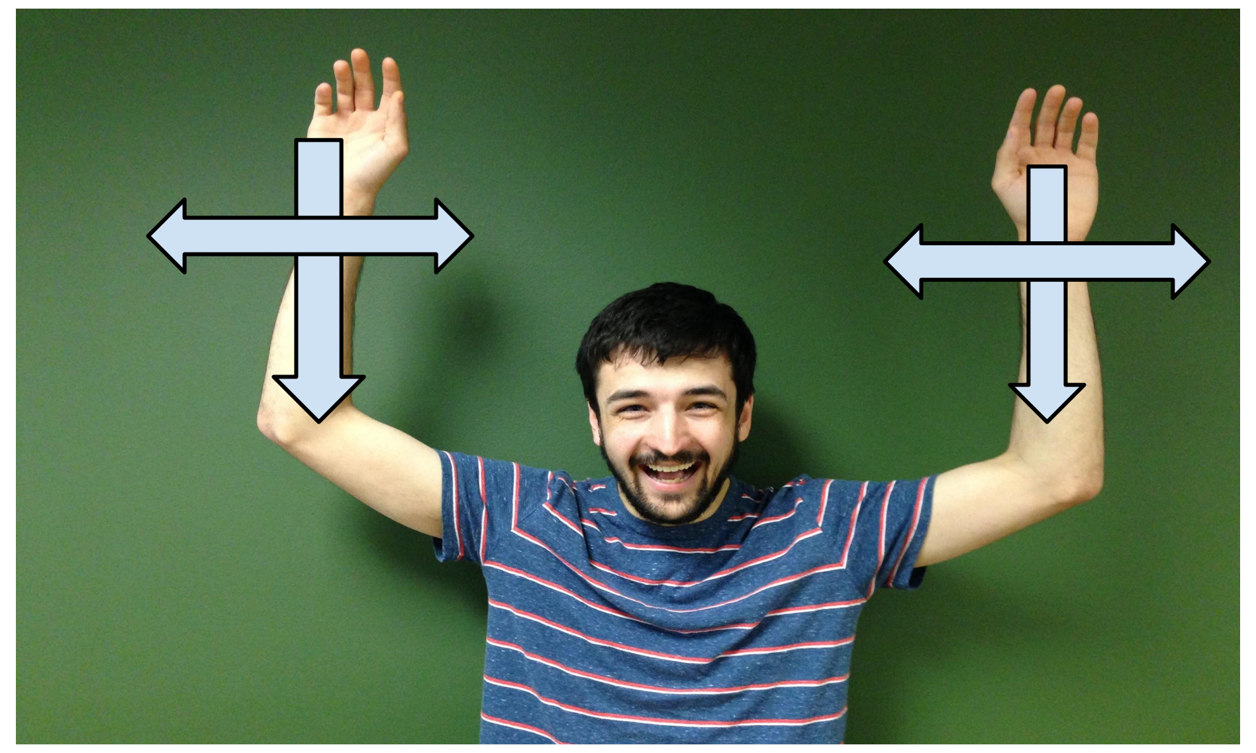
Kinect

The game supports the original Kinect for Windows/Kinect 360. The SDK used was Kinect for Windows v1.8.









Right Hand: Camera/Aiming

Left Hand: Movement - Left/Right Strafing

Jump: Raise Left Elbow above Shoulder.

Players move forward when the left hand is forward and move backwards when the hand is upright.
The camera moves up when the right hand is in the upright position and moves down when the hand is forward.

The commands are handled by first taking the distance between your elbow and wrist, then doing ratio checks on it to determine if your hand is forward or upright. The position in space does not matter except when checking if the elbow is higher than the shoulder. Everything else is based on ratios of the distance between two points of the body.

Networking

Photon Unity Network (PUN) was used to handle connecting remote clients via a Cloud based server system. Remote Procedural Calls (RPC) were used to communicate between clients and keep the games synced. This is used to have all the clients making identical calls on their local copy of game objects. Clients can either join existing rooms or create a new room and become the master client. When a player joins a room they are synced with the master client and player's load scenes when the master client loads. Objects are instantiated over the network and each player keeps a local copy that continually gets data from the Cloud but only your player has scripts activated so it is the only character that is controlled by you.

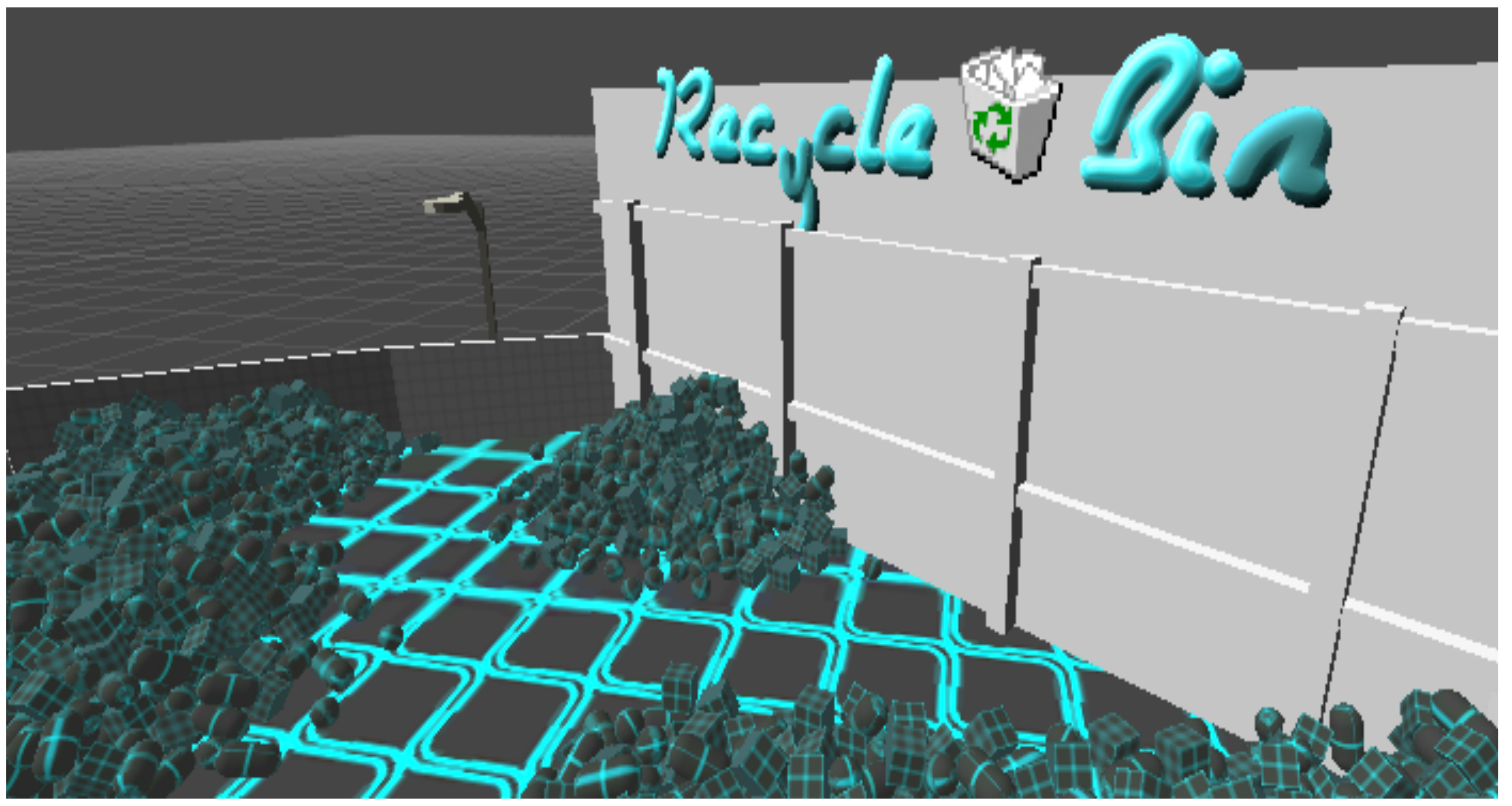
Bullets

The bullets are made by having a spherical game object with a rigidbody attached to handle movement and collision detection is handled by raycasting. The raycast checks what it collides with and allows the bullet to pass through pickups, damage players and destructible surfaces, and be destroyed when impacting the map itself.

Artificial Intelligence

AI is used in the single player mode and each level has unique AI functionality. The mice on Rosso's Moon Landing use character controllers to move and swarm towards the player by tracking the player's current position. The fox in Murasaki's forest uses a random integer generator to choose a location to move to and then rotates and moves behind the stump. In Tangelo says the commands are generated by checking a dictionary of commands and randomly choosing one to ask.

Level Building



Multiplayer levels were handmade using Prototype by ProCore.

Rosso's Moon Landing is procedurally generated by creating a grid of cubes and removing cubes with an algorithm to create a level with obstacles.