Heartbeat Horror

Mitchell Neal and Jacob Brown
Team Polsky

maneal@uchicago.edu
jacobpeterbrown@uchicago.edu

Background

Problem

Games are lacking in their ability to detect the quality of a player's game experience, and to respond to it

Idea

A game that senses the player's involuntary responses during the game, and is able to change the gameplay experience accordingly in real time.

Our Idea: Your Heartbeat is Your Health Bar

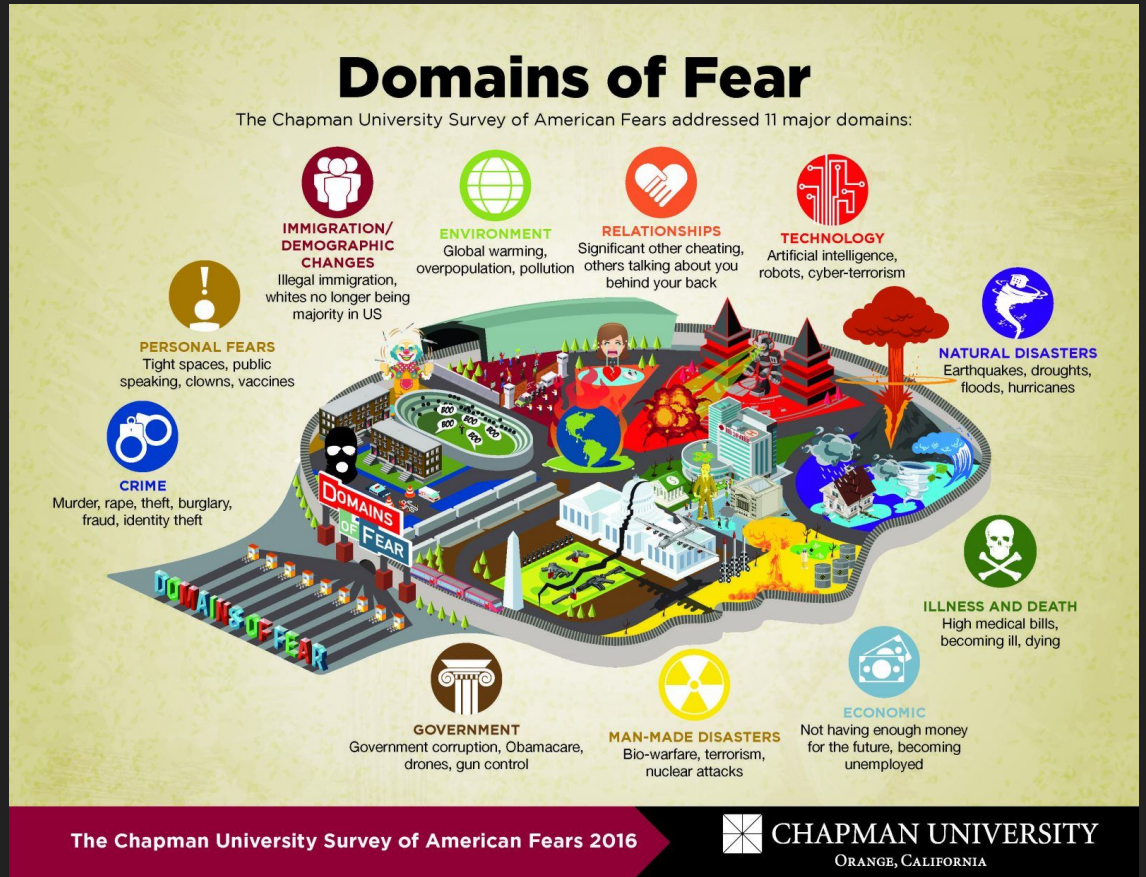
- The game senses the player's heart rate, and the player loses if their HR gets too high
- The game can change based on what it detects scares the player most

Basic Description

- In the game, you are taken through a series of different scenes. In this live demo, we will only be showing the first scene.
- This game represents a proof-of-concept of the technology, so movement through the scene is scripted; the only motion you control is head motion
- On the menu screen, we estimate a baseline heart rate. In the game, if your heart rate rises 40% above this baseline heart rate, you lose the game.
- We have 3 pieces of software involved in monitoring your heart rate. 1) an app on the watch that reads heartbeat data. 2) a background app on the phone that takes the data from the watch and broadcasts it to the Unity game 3) a unity plugin that listens for HR data

Live Demo!
Volunteers?

Background Research: Hierarchies of Fears



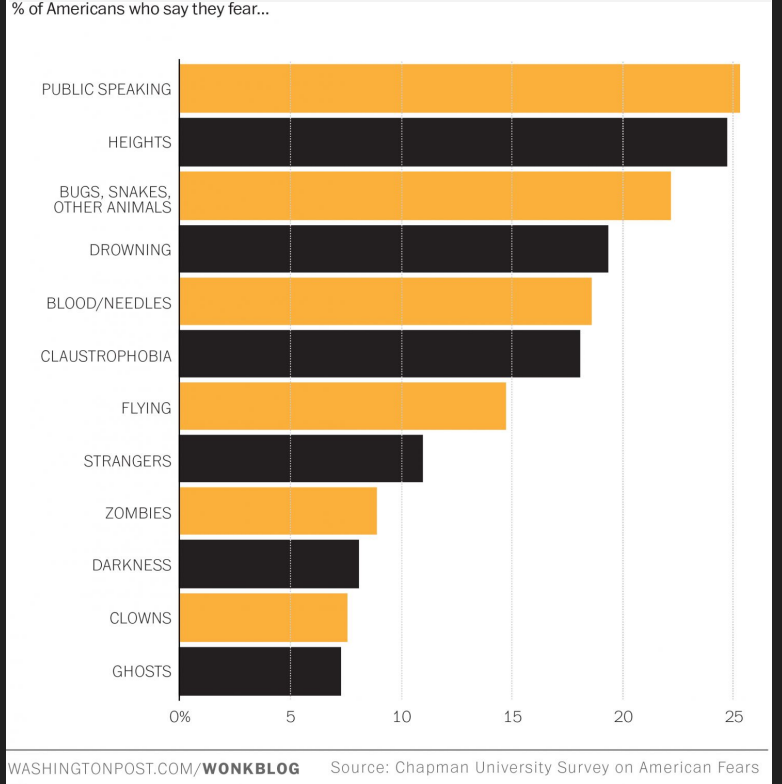
Background Research- Hierarchies of Fears

Top 10 Fears

Fear	Fear Domain	% Afraid or Very Afraid
Corrupt government officials	Government	60.6
Terrorist Attack	Manmade Disasters	41
Not having enough money for the future	Economic	39.9
Terrorism	Crime	38.5
Government restrictions on firearms and ammunition	Government	38.5
People I love dying	Illness and Death	38.1
Economic/financial collapse	Economic	37.5
Identity theft	Crime	37.1
People I love becoming seriously ill	Illness and Death	35.9
The Affordable Health Care Act/Obamacare	Government	35.5

Notice: Most of these are not actionable in a simple video game!

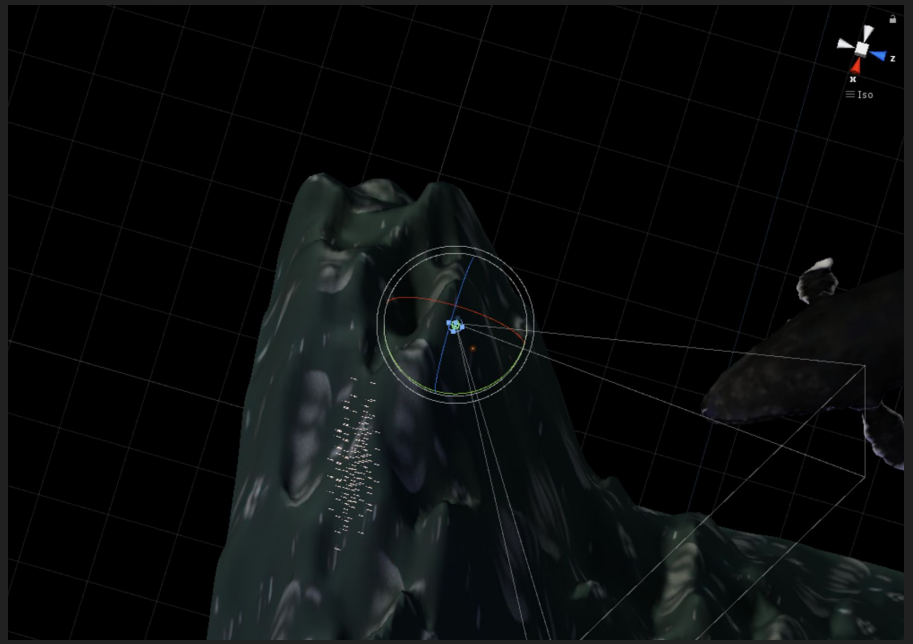
Top 10 actionable fears

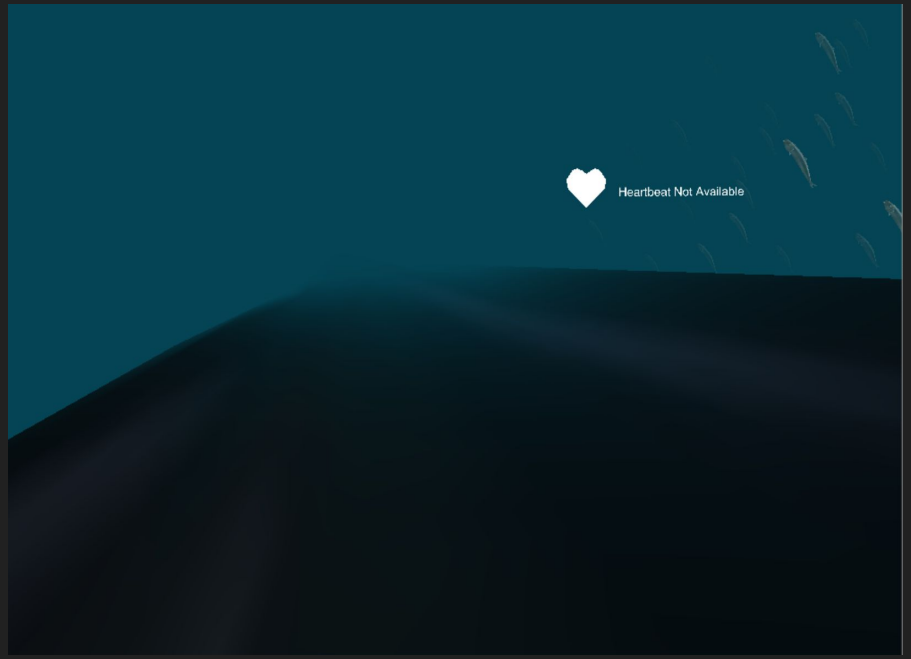


Scenes Not Shown in the Live Demo

Underwater Sea Monsters Scene:

Plays on fear of water/drowning, vague shapes drift in and out of peripheral vision





Scenes Not Shown in the Live Demo

Hospital Scene:

Fear of blood/hospitals, darkness. Long hallway directs player's vision forward, leaving the periphery for surprises

Scenes Not Shown in the Live Demo

Urban Rooftop Scene:

Rooftop setting plays on fear of heights

Fear Manager

- Object in the scene that changes gameplay based on monitored response
  - Responses considered: your heartbeat, the speed of your head movement, and what objects your eyesight lingers on.
- In general, it uses these responses to determine whether you are scared of individual objects in the game world.
  - Once deciding you are scared of an object, it will them increase or decrease the frequency of this object in the game world, depending upon whether the code is set to “easy” or “hard” mode.
- The algorithm that determines if are you are scared of any specific object is based upon the behavior of a neuron firing an impulse.

Fear Manager Code Snippets

```
        //SPHERECASTING
        if(Application.loadedLevelName == "forest"){

            Spherecast(forward, hit);


        }
}

void Spherecast(Vector3 forward,RaycastHit hit)
{

    CharacterController charCtrl = GetComponent<CharacterController>();

    if (Physics.SphereCast(forward, charCtrl.height / 2, transform.forward, out hit, 5))
    {
        if (hit.collider.gameObject.CompareTag("littlespider"))
            {

            if (GameObject.Find("InputManager").GetComponent<InputManager>().checkForClick())
            {

                GameObject.Find("Heartbeat Manager").GetComponent<FearManager>().scareoflittlespiders += .5;
                mLastObjectHit = hit.collider.gameObject;

            }
            }

        if (hit.collider.gameObject.CompareTag("bigspider"))
            {

            if (GameObject.Find("InputManager").GetComponent<InputManager>().checkForClick())
            {

                GameObject.Find("Heartbeat Manager").GetComponent<FearManager>().scaredofbigspiderscares += .5;
                mLastObjectHit = hit.collider.gameObject;

            }
            }

        if (hit.collider.gameObject.CompareTag("flatlegs"))
            {
```

```
RaycastHit hit = new RaycastHit();
comparison2 = comparison;
comparison = forward;
forward = GameObject.Find("Reticle").GetComponent<Reticle>().transform.TransformDirection(Vector3.forward);


if(Math.Abs(forward.x - comparison2.x) > 5 || Math.Abs(forward.y - comparison2.y) > 5){

    GameObject.Find("Heartbeat Manager").GetComponent<FearManager>().scareofspiders += .5;
}
```

```
    if(health != null)
    {
    //health.GetComponent<TextMesh>().text = input.ToString();
    }
    //moderately scared
    if(heartbeat > restingheartrate * 1.2)
    {
        RenderSettings.fog = true;
        RenderSettings.fogColor = Color.red;
        RenderSettings.fogDensity = .05f;
        AudioSource audio = this.GetComponent<AudioSource>();
        audio.Play();

        //if the person gets scared of spiders, add them to the next
        scene
        if(Application.loadedLevelName == "forest"){

            //getting players x coordinate location
            float x = GameObject.Find("guy with flashlight")
                .transform.position.x;

            //good bet that they are scared of spiders
            scaredofspiders += 1;

            if(x < 50)
            {
                scareoflittlespiders += .5;
            }

            if(x > 58 && x < 68)
            {
                scaredofbigspiderscares += .5;
            }

            if(x > 68 && x < 74)
            {
                scaredofemptiness += .5;
            }

            if(x > 74 && x < 97)
            {
                scaredoflatlegs += .5;
            }

            if(x > 98 && x < 120)
            {
                scaredofemptiness += .5;
            }

            if(x > 120)
            {
                scaredofbigspiderscares += .5;
            }
        }
    }
```

Android Wear App/Unity Plugin Code Snippets

Wear App Pseudocode

```
public class MainActivity extends Activity implements
        DataApi.DataListener, GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener, SensorEventListener {

    private GoogleApiClient googleClient;
    private SensorManager mSensorManager;
    private Sensor mHR;

    protected void onCreate(Bundle savedInstanceState) {
        //PseudoCode:
        Initialize(mSensorManager);
        Initialize(googleClient);
    }

    public void onSensorChanged(SensorEvent event) {
        if (event.sensor.getType() == Sensor.TYPE_HEART_RATE) {
            PutDataMapRequest putDataMapRequest = PutDataMapRequest.create();
            putDataMapRequest.getDataMap().putString("message", event.hrdata());
            putDataRequest.setUrgent();
            PendingResult<DataApi.DataItemResult> pendingResult = Wearable.DataApi.putDataItem(googleClient, putDataRequest);
        }
    }
```

Mobile Listener/Forwarder Pseudocode

```
ublic class GoodIntentions extends Service implements
        GoogleApiClient.ConnectionCallbacks, DataApi.DataListener,
        GoogleApiClient.OnConnectionFailedListener{

    private GoogleApiClient googleClient;

    public void create() {
        //Pseudocode:
        Initialize(googleClient);
        googleClient.connect();
    }

    @Override
    public void onDataChanged(DataEventBuffer dataEvents) {
      //Get data from dataEvent
      DataItem item = event.getDataItem();
      DataMapItem dataMapItem = DataMapItem.fromDataItem(item);
      String hr = dataMapItem.getDataMap().getString("message");
      //Create Intent to broadcast HR data to Unity App
      Intent intent = new Intent();
      intent.putExtra(Intent.EXTRA_TEXT, hr);
      sendBroadcast(intent);
    }
```

Unity Listener Pseudocode

```
public class Receiver extends BroadcastReceiver {

    private static Receiver instance;
    public static String text = "";

    @Override
    public void onReceive(Context context, Intent intent) {
        String message = intent.getStringExtra(Intent.EXTRA_TEXT);
        if(message != null) {
            text = message;
        }
    }
```

Challenges

- To access deeper Android features such as messaging and listener services, we needed to create a Unity plugin in Android Studio

Sense HR > Send Data from Watch > Listen for Data on Phone > Send to Unity Plugin > **Use in Unity**

- What we wanted to do here isn't very well documented
- We found a few startups devoted to sensing biometric data for VR games

Next Steps

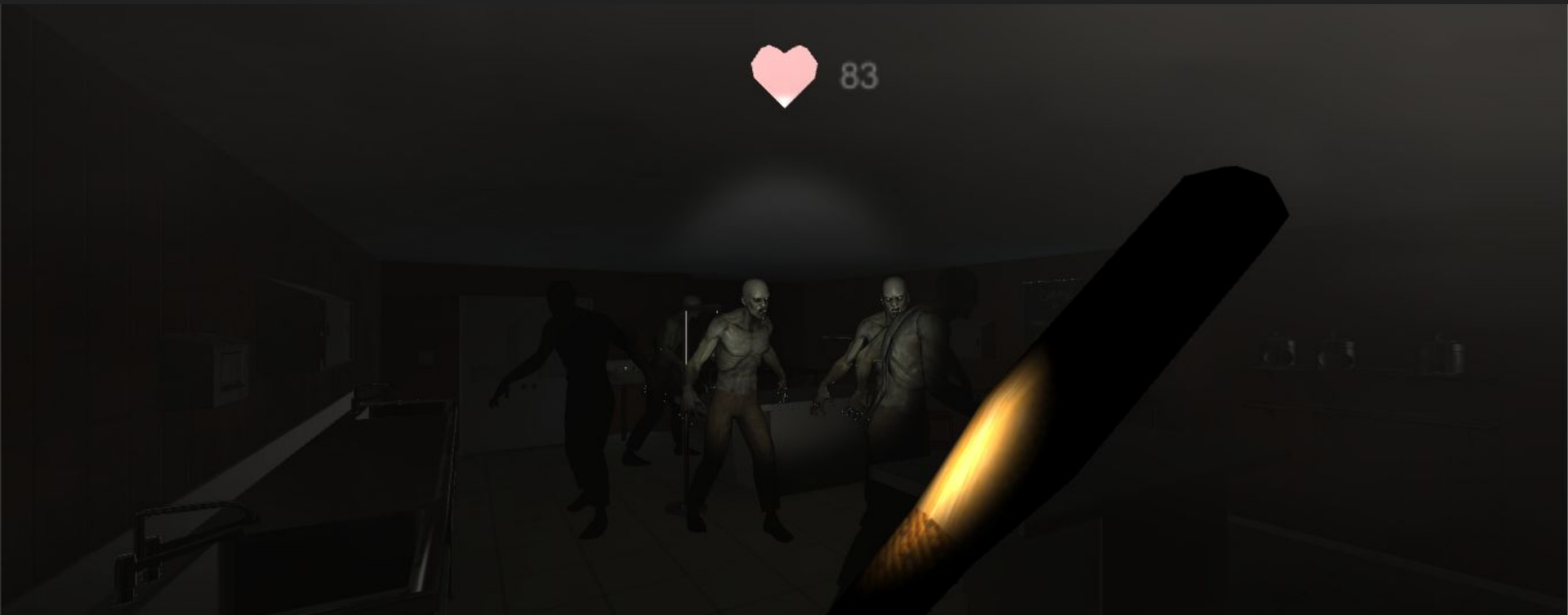
- Improve our game to include more player-directed movement and control
- Adding the ability to monitor verbal data
- More play testing

- Do even more with involuntary player feedback

More Advanced Scenes We Have Planned Are...

Future Additions to the Game

- Level where zombies can ‘smell fear’ and only attack if you’re afraid
- Plays on fear from anticipated danger



Future Additions to the Game

- Crossing a narrow bridge
- Camera shakes if your heart rate is too high, simulating impairment from fear

Thanks!

Questions?