Lost Frequencies

By Brigitte Parnham and Zala Sesko

PROJECT DESCRIPTION

Lost Frequencies is an interactive web-based program representing a speculative reality. Contrived as an explorative game, it's focus is on the concept, narration and user experience. Upon starting the game, the player enters a world stripped of language where the main form of communication are light and sounds emerging from interaction between the characters' auras. These are assigned to people as they are born in the world, meaning each person is characterised by the colour of their aura and the sound accompanying it, both of which are clearly visible to the individual and everyone around them. Consequently, the world becomes rather "transparent," since people's true intentions are perceived from the onset, leading to less miscommunication in comparison to our reality. Therefore, the project's purpose is to get the user to question how they communicate with others in the real world, as it is usually affected by superficial factors (gender, race, sexuality, etc.).

Due to the conceptual nature of Lost Frequencies, there are no concrete goals that the player is required to achieve, leading to the project being a "non-game" game, which allows the user to freely explore the world while reflecting on the idea of the narrative. The conceptual background is provided right at the start to ensure the player understands the project.

The key elements are therefore comprised of graphics and synthesised sounds such as colourful animations (representing auras) and sine waves of various frequencies. While the main feature is exploration and experience of changes of colour and sound, additionally the player is able to "step" into the game through the use of camera input, computer vision and image processing – namely, they are able to see their aura radiating around them as augmented reality and even take a snapshot of it. By designing the project in the style of retro games, yet keeping the environment realistic, we hope to achieve the feeling of familiarity infused with peculiarity.

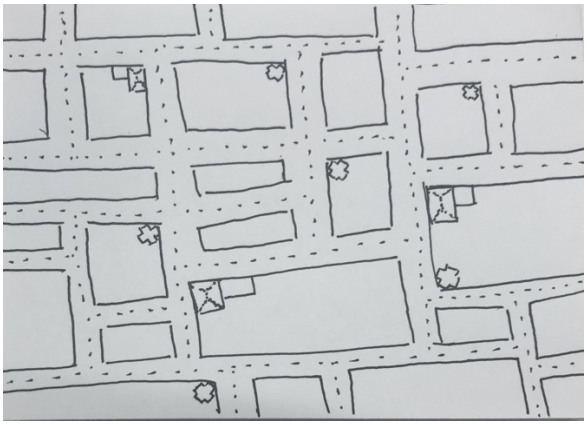


Figure 1: Map sketch

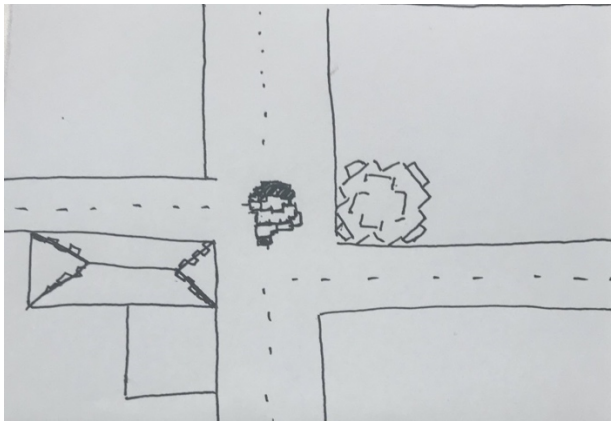


Figure 2: Player view sketch

The above sketches are representing the initial idea depicting the map of the world. The left on illustrates the whole world, showing the placement different objects such as houses, roads, land and trees, whereas the sketch on the right shows how the user sees the world at any point in time – having limited perception to further correlate with reality.

The following sketch illustrates what the user sees – their aura – when they are in the camera mode of the game. Throughout the project these initial representations served as the foundation underpinning succeeding choices. They were built upon and considered at every step, however, were not altered substantially.

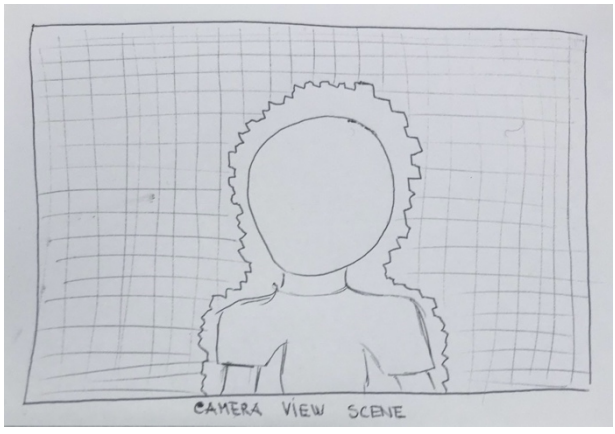


Figure 3: Camera scene sketch

AUDIENCE

We hope to show the audience how social interactions would change with different 'senses' (referring to an aura as a sense). By stripping away verbal communication, we create a universal experience as the 'language' of sound is commonly understood. The user thus must learn to navigate the world relying on other perception abilities.

Consequently, by neglecting the concept of personality and defining each person based on their aura alone, the user should discover the direct reflection of the real world in which people often judge each other based on stereotypes/their outward appearance.

When interacting with other characters, their auras merge, creating different colours and sounds. This is not to show that some interactions are good or bad, but simply that they vary between different aura combinations. Similarly, in real life people perceive their interactions subjectively even though they might be objectively nearly identical. To mirror this, the merging of the same aura combinations always produces the same result, however its interpretation is subjective and varies depending on the player. Some might experience in a more pleasant, while others in a more unpleasant way.

Upon getting feedback about the initial versions of the project, we quickly discovered that our intentions were not immediately clear to the audience. Tackling this issue, we made sure to include detailed instructions, relating them directly to the concept, in later versions. Moreover, although we intended to make the sounds just on the verge of discomfort in order to emphasise their influence, it seems we overestimated the player's capacity to not be overly disturbed by them, leading them to not enjoying the game as much. Wanting to keep our audience engaged, we thus changed the sounds accordingly, further testing them with the users. Similarly, engagement changed throughout the development of the camera feature. In the beginning states, its meaning was not evident to the audience, however, they seemed to enjoy the feature once it was finalised.

Figure 4: User testing

Figure 5: User testing

TOOLS, EQUIPMENT AND KNOWLEDGE

During the making of our project we have used different tools and knowledge to get the outcome which we strived to achieve. Using p5.js we have created a clean graphical and interactive experience, with other libraries that work seamlessly with p5.js. Wanting to have a multiple scene game so that the user could switch between the game world and the video capture scene we used p5.SceneManager. To enhance our graphics, we have used p5.play which has animated the images of the players and the others to create smooth movement. Achieving the realistic aura in the video scene we have used ml5.js a machine learning algorithm library. As well as using MaxiLib to synthesise the sounds for the players and others.

Code samples

The following code shows how we have used p5.play to animate our players in the seamless animation.

```
this.animate = function()
{
    if(this.down)
    {
        //walk down animation
        animation(walkDown_png, this.xpos, this.ypos);
    }
    else if(this.up)
    {
        animation(walkUp_png, this.xpos, this.ypos);
    }
    else if(this.left)
    {
        animation(walkLeft_png, this.xpos, this.ypos);
    }
    else if(this.right)
    {
        animation(walkRight_png, this.xpos, this.ypos);
    }
    else
    {
        //still
        image(still_png, this.xpos - still_png.width/2, this.ypos -
        still_png.height/2);
    }
}
```

Figure 6: Player animations

This is how we implemented ml5.js and used it to find the points on and around the face, while connecting them to the noise circle around the head.

```
// A function to line around face and then aura shape
this.AuraBackground = function(startColour, endColour){
    // Loop through all the poses detected
    for(var j=0; j<numCircles; j++){
        for (let i = 0; i < poses.length; i++) {
            //maps a colour between the two start and end for each circle
            var mapColour = map(j, 0, numCircles-1, 0, 1);
            if(i == 0 ){
                var personColour = lerpColor(startColour, endColour, mapColour);
                fill(personColour);
            }
            else{
                var r = othersR[i];
                var startOther = start[r];
                var endOther = end[r];
                var otherColour = lerpColor(startOther,endOther,mapColour);
                fill(otherColour)
            }

            // For each pose detected, loop through all the keypoints

            let pose = poses[i].pose;
            //original points
            let nose = pose.keypoints[0];
            let leftEye = pose.keypoints[1];
            let rightEye = pose.keypoints[2];
            let leftEar = pose.keypoints[3];
            let rightEar = pose.keypoints[4];
            let leftSholder = pose.keypoints[5];
            let rightSholder = pose.keypoints[6];

            //using the points that the ml5.js works out we create are own points
            let faceWidth = leftEar.position.x- rightEar.position.x;

            let foreheadPointX = nose.position.x;
            let foreheadPointY = nose.position.y - faceWidth*(4/5);

            let foreheadPointX1 = rightEye.position.x;
            let foreheadPointY1 = rightEye.position.y - faceWidth/2;

            let foreheadPointX2 = leftEye.position.x;
            let foreheadPointY2 = leftEye.position.y - faceWidth/2;

            let foreheadPointX3 = rightEar.position.x +10;
            let foreheadPointY3 = rightEar.position.y - faceWidth*(2/5);

            let foreheadPointX4 = leftEar.position.x -10;
            let foreheadPointY4 = leftEar.position.y - faceWidth*(2/5);
```

```
            let distBXaX1 = foreheadPointX - foreheadPointX1;
            let distBYaY1 = foreheadPointY1 - foreheadPointY;

            let foreheadPointX5 = foreheadPointX -distBXaX1/2;
            let foreheadPointY5 = foreheadPointY + distBYaY1/2;

            let distBXaX2 = foreheadPointX2 - foreheadPointX;
            let distBYaY2 = foreheadPointY - foreheadPointY2;

            let foreheadPointX6 = foreheadPointX + distBXaX2/2;
            let foreheadPointY6 = foreheadPointY - distBYaY2/2;

            let distBX1aX3 = foreheadPointX1 - foreheadPointX3;
            let distBY1aY3 = foreheadPointY3 - foreheadPointY1;

            let foreheadPointX7 = foreheadPointX1 - distBX1aX3/2;
            let foreheadPointY7 = foreheadPointY1 + distBY1aY3/2;

            let distBX2aX4 = foreheadPointX4 - foreheadPointX2;
            let distBY2aY4 = foreheadPointY4 - foreheadPointY2;

            let foreheadPointX8 = foreheadPointX2 + distBX2aX4/2;
            let foreheadPointY8 = foreheadPointY2 + distBY2aY4/2;

            let distBX3aRE = foreheadPointX3 - rightEar.position.x;
            let distBY3aRE = rightEar.position.y - foreheadPointY3;

            let foreheadPointX9 = rightEar.position.x + distBX3aRE/2;
            let foreheadPointY9 = rightEar.position.y - distBY3aRE/2;

            let distBX4aLE = leftEar.position.x - foreheadPointX4;
            let distBY4aLE = leftEar.position.y - foreheadPointY4;

            let foreheadPointX10 = leftEar.position.x - distBX4aLE/2;
            let foreheadPointY10 = rightEar.position.y - distBY4aLE/2;

            let foreheadPointX11 = rightEye.position.x;
            let foreheadPointY11 = rightEye.position.y + faceWidth*(4/5);

            let foreheadPointX12 = leftEye.position.x;
            let foreheadPointY12 = leftEye.position.y + faceWidth*(4/5);

            let foreheadPointX13 = foreheadPointX4;
            let foreheadPointY13 = foreheadPointY4 + faceWidth*(4/5);

            let foreheadPointX14 = foreheadPointX3;
            let foreheadPointY14 = foreheadPointY3 + faceWidth*(4/5);

            noStroke();
            beginShape();
```

Figure 7: Face tracking

… connects all points around face …..

```
            //have to go from 180 to 0 to get the semicircle effect and backwards so that the rightshoulder meets with the first point and goes round to the left and
            closes
            for(var angle =180; angle>0; angle-=1.0)
            {
                vertex(points[j][angle][0]+nose.position.x, points[j][angle][1]+height+20);
            }
            endShape(CLOSE);
        }
    }
}

//gets the noise to add to each point
this.CircularNoise = function(scale){
    this.offsetX = random(100000);
    this.offsetY = random(100000);
    this.offsetZ = random(100000);
    //the bigger the scale the less smooth the shape
    this.scale = scale;

    //using the frameCount it means the points move
    this.getNoise= function(radian, time){
        var r = radian % TWO_PI;
        //makes sure that the r doesnt go bellow 0
        if(r < 0.0){
            r += TWO_PI;
        }
        //returns a noise value which is worked out from the x and y and time
        return noise(this.offsetX + cos(r) * this.scale,
                     this.offsetY + sin(r) * this.scale,
                     this.offsetZ + time);
    }
}
```

Figure 8: Face tracking pt.2

Inspiration

Figure 9: Inspiration mood board

1. https://www.spriters-resource.com/game_boy_advance/pokemonfireredleafgreen/sheet/3768/

2. https://grafxkid.tumblr.com/post/181903236134/im-making-extremely-good-progress-on-the-world-of

3. https://binbin.vn/manor-town-game-map-of-pokemon-pets-route-id-13-zone-normal-295416.html

4. https://imgur.com/a/un3va

5. https://www.shutterstock.com/image-illustration/seamless-repeating-pattern-472817071

6. https://www.gamedevmarket.net/asset/rpg-slime-monster-1673/

7. https://www.dreamstime.com/animated-slime-character-sprites-creating-fantasy-rpg-adventure-video-games-image116939692

8 – 10. Christina Lonsdale, Radiant Human, Available at: http://www.radianthuman.com

While we have taken inspiration from many games and ideas throughout the process of making our project, the following are the main ones that have influenced us. For the appearance of the game we have taken stimulus heavily from 8bit retro (pixel art) games and even the likes of early Pokémon games. The way in which are auras look have been informed by Susan Hiller, who has created aura like forms with her use of colour blotched around the sitter's face, which looks like it emits off them. Another artist who has expressed auras in a different way is Christina Lonsdale, who uses a Guy Coggins camera, which supposedly captures people's aura or otherwise known as the electromagnetic field surrounding the body. We have tried to build a bridge between the repeatedly tested but never proven existence of auras and well researched physical phenomena (light, sound). Even though the connection between light and sound is often mentioned in reference with synaesthesia, Lost Frequencies extends this knowledge into a speculative and even paranormal world.

Figure 10: Susan Hiller's depiction of aura-like forms in Homage to Marcel Duchamp: Auras, 2008

Many have tried to find a direct connection between light and sound by juxtaposing the spectrum of audible frequencies and the spectrum of visible frequencies even proposing prototypes of various 'hyper-instruments', as in a study done by André Rangel Macedo, 2009, and other digital prototypes, such as mentioned in Mengucci et al., 2012, paper, however the direct connection between the two seems to be outside of the meaningful range for humans to hear or see. For the purposes of the game, the so-called Sonochromatic Music Scale was be used, which divides the octaves into 360 notes, each corresponding to a specific degree of the colour wheel, simplifying the concept and mapping the result to a range humans are able to perceive.

SONOCHROMATIC MUSIC SCALE (basic 12/360)		
	Rose	**E**
	Magenta	**D#**
	Violet	**D**
	Blue	**C#**
	Azure	**C**
	Cyan	**B**
	Spring	**A#**
	Green	**A**
	Chartreuse	**G#**
	Yellow	**G**
	Orange	**F#**
	Red	**F**

Figure 11: Sonochromatism. Wikipedia (2020). [online] https://en.wikipedia.org/wiki/Sonochromatism

DEVELOPMENTAL PROCESS

Prototype 1

As the first prototype was intended as a broad testing of our idea, implementation of only a few significant elements was needed. Thus, we focused on the interaction between the player and other characters by making sure that the colours changed appropriately. Neglecting the idea of using GIFs for auras, we used particle systems to do so and kept the overall style minimalistic.

For the aura aesthetic in the video capturing we decided to pixilate the video feed and layer a tinned rectangle over. Hoping this would slightly distort the image to create a flickering effect with a change in colour to imply that the user has an aura.

We created sounds for each aura following the Sonochromatic music scale which links specific degrees on the colour wheel with the relevant Hz, in theory each colour has a sound. Using a sinewave at that Hz related to that colour for each aura variation. When the character interacted with another in the game their sound waves would mix to make their interaction sound.

Users didn't respond much to the first design of the camera as they didn't understand what it was meant to represent. A main point that users brought up was that they didn't see what the pixilation added to the design but rather detracted from the look and theme.

This first version of sounds didn't go down, with users, very well. They understood our theory but suggested using noise or creating more of a harmonious sound.

On the other hand, many commented they liked the look of the game and when asked whether they'd prefer a more realistic style or an abstract one, some were keen on the latter.

Prototype 2

Taking on board the feedback from the last prototype we decided to scrape the pixilation and investigate other ways to represent an aura in the video. This is when we came up with the idea of tracking the outside of the user and having the aura surround the user and not just be an overlay. At this stage we tried tracking.js and opencv.js not getting useable results from either of them.

We tried various combinations of noise and other waves, and eventually decided to change the players sound to an additive synthesis of brown noise and a sine wave and the sounds of the others to Perlin noise with sine waves. Producing more pleasant to listen sounds while maintaining variation between combination of auras.

The opinions from our users were mixed when it came to the sound as they still didn't find the sound a fully enjoyable experience, but we felt that this was part of the experience as it helps convey the overall feeling we had wanted to invoke.

Prototype 3

Looking at new libraries for computer vision that worked with p5.js we found ml5.js which is an approach to creating and exploring AI in the browser, built on top of TensorFlow.js. Instead of finding the edge of the face or just detecting the face, which

the other libraries had done, this code found points on the body and connected them. We had finally got a result that successfully tinned the background with the aura while leaving the head and shoulders as the original feed.

Regarding graphical elements, we started to develop the background (the world). Being fond of the idea of the simplistic style, we made it grey with several rectangles and other shapes representing houses, land and trees. Furthermore, we started to play with various character designs – we considered having a pixelated human character as those in retro games, an abstract figure or a non-pixelated human character and whether they would be animated while moving or would they just switch between views (front view, left side view,…).

Positively, most of the users said that this camera result did look like there was a force around them but not specifically what this was, one calling it a 'shroud'. When asked how we could improve the result one user suggested that we shouldn't fill the whole background but rather create a 'aura' like shape around the user, which is filled.

After testing the graphical elements, we decided on a 'blob-like' structure, which would emphasize the fact that the project is a representation of a speculative reality and keep the user engaged as it was not just another real-world character, but something different. That approach also allowed for easier display of the various auras, as each 'blob' would be of the colour of their aura, making the latter the truly distinctive feature of each individual character, further emphasizing our project vision. The world, however, turned out to be a bit too abstract and thus not contributing to having realistic elements. Furthermore, it did not look as developed as it could have been.

Prototype 4

At this point in the camera screen we tidied up the points around the face making the appearance smoother and more natural. We also looked into ways to create an aura like force shape around the person, this is where we came up with the idea of a layered noise circle which connected to the points around the face and had a flowing movement to it. In a previous user feedback one of the users had tried to see if having two of them in the video would change the auras, at the time we didn't have anything like this working but this had given us the idea to implement it later. The final development of the video scene was to add a camera button which meant the user could save a selfie of themselves with their aura.

Moving away from the idea of a simple abstract background, we decided to go back to the idea of a pixelated version as if was more important that the user could tell they're

in a (speculative) world by keeping some of the elements realistic. Thus, our main source of inspiration was the style of old Pokémon and Gameboy games, making the result similar to pixel art while not being too distracting due to the wide variety of elements comprising it. Turning down the saturation also contributed to the effect.

To keep computational power required for displaying the background on a minimum, we concluded it be best to draw it a separate software (Adobe Illustrator) and then importing the image.

Figure 12: A still from the final background

At this point we had hoped that the sound was working but with the developed game it seemed to glitch in the form it was at the moment, so we decided to change it to brown noise with a sine wave for both the player and the others. This created a smoother additive synthesis than before.

Before we got the noise circle points to move around the head, users said that the shape around the head looked more like a hoodie or mass of hair. This was worrying, as we wondered if this would mean that our idea wouldn't convey as an aura. However, once we got the whole 'aura' moving in a fluid motion, users recognised it as an aura or other worldly force around them.

Prototype 5

Throughout our prototype's users had to click keys *1* and *2* on their keyboard to go between different screens. This didn't make much sense and broke the flow of the game quite a bit. Thus, we decided to implement an "*access*" feature to some houses, allowing the user to switch between different screens in the game more intuitively. By adding question buttons on each screen, the users could find out more information about the screen they were using. A globe button was implemented, too, which would bring a user back to the main game. Lastly, we added more information to the start screen to introduce the user to the concept right at the beginning and even an additional information page (accessible through a house) where they could read more about the whole project.

Figure 13: Development mood board

CHALLENGES

Sound

The sound in the project was a constant debate for us as the under-pinning idea of our project was the relation to sound and light, but users often struggled with our sounds. This meant that we tried many ways to change it to something more enjoyable, however

by doing this we lost the 'uncomfortableness' that the sound often created. Nevertheless, we felt that these emotional reactions to the sound were necessary in showing a direct link to overseen uncomfortableness that we have in everyday life.

Camera

Getting to grips with new libraries was one of the challenges we faced while doing the project, as we had to learn how they worked and integrated with p5.js. Throughout the making of the video scene we tried many different libraries and ways in which we could draw the aura without it being a fixed shape that had no relation to the face. Struggling to get many of these libraries to work in the way we wanted them to, it was hard to keep developing the idea as we hit many dead ends. However, by sticking to it and learning multiple ways round the problem we ended up with a result that we are very happy with.

When we came up with the idea of having layered noise around the head, we struggled to connect the points from the noise circle to the face. The first problem was that the original noise circle code didn't store points (so couldn't connect) and when adapting it to do so, it meant that the points around the circle didn't animate. After changing the code to a version that took each point and updated them each run of the draw while using the *frameRate* to animate, we hit another problem. The translation and rotation which we were using meant that either the points around the face moved or the noise circle points stuck in the left corner. By not only finding the points for 180 degrees but for 360 and then drawing the points from 180 to 0 we got the top of the circle drawn and then adding the nose x position and height of the canvas we got the animation in the right position moving with the face.

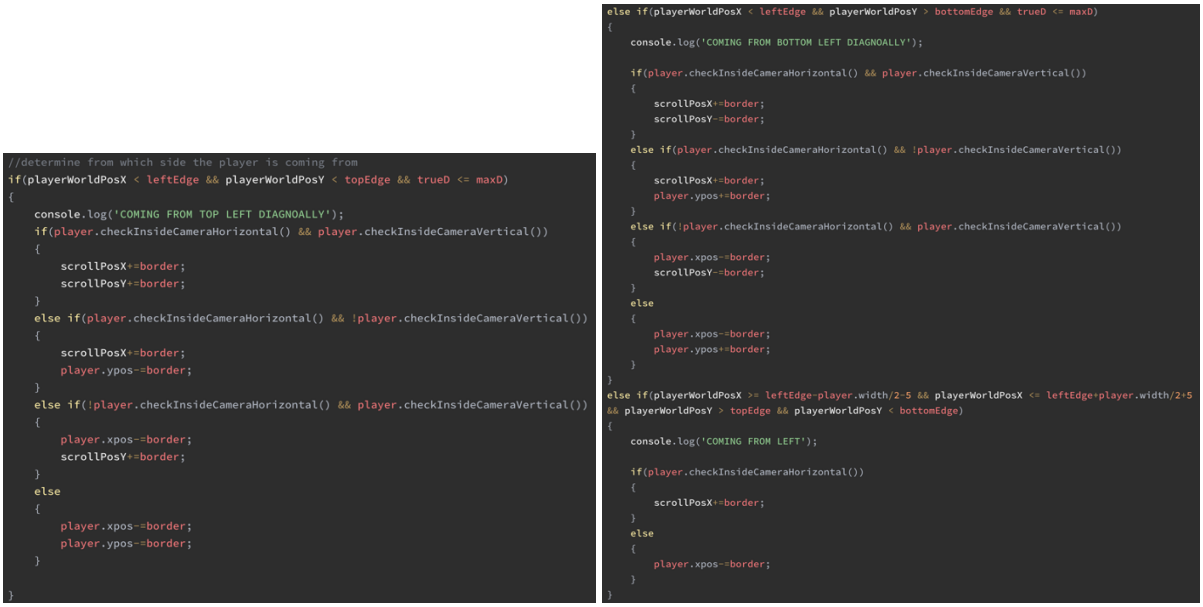
Movement constraints

Another challenge we faced was constraining the player's movement, so they were only allowed to move inside the world without being able to move on top of houses. After a discussion with the TAs it seemed the most efficient to move the player a few pixels back when they would collide with a house. Our first approach of constraining the player's movement consisted of calculating the distance between the centre of the house and the player and comparing it to the sum of the house's diagonal and player's radius (considering it was displayed as a circle at that point). If the distance was anything smaller, the player would be pushed back depending on the side they were coming from.

```
//old - doesn't work
    if(playerWorldPosX < leftEdge && playerWorldPosY < topEdge)
    {
        if(trueD <= maxD)
        {
        console.log('COMING FROM TOP LEFT DIAGNOALLY');
        scrollPosX+=border;
        scrollPosY+=border;
        }
    }
    else if(playerWorldPosX < leftEdge && playerWorldPosY > topEdge && playerWorldPosY <
    bottomEdge)
    {
            console.log('COMING FROM LEFT');
            scrollPosX+=border;
    }
    else if(playerWorldPosX < leftEdge && playerWorldPosY > bottomEdge)
    {
        if(trueD <= maxD)
        {
        console.log('COMING FROM BOTTOM LEFT DIAGNOALLY');
        scrollPosX+=border;
        scrollPosY-=border;
        }
      }
    else if(playerWorldPosX > leftEdge && playerWorldPosX < rightEdge && playerWorldPosY
    > bottomEdge)
    {
        if(trueD <= this.height/2 + player.width/2 + 5)
        {
            console.log('COMING FROM THE BOTTOM');
            scrollPosY-=border;
        }
    }
```

Figure 14: Initial house collision checking

However, that approach turned out to be too simplistic, as it only worked if the houses were squares and even then only on the corners, while on the sides the player couldn't go as near as we had envisioned. After a lot of trial and error, we got it too work, establishing that checking was needed to be done separately for each of the possible ways of approaching (left, right, bottom, top, each of the four corners). However, that was still not the final solution. This approach only worked if the player and the house were both inside the 'camera world' i.e. the area in which the camera moves, and the player stays in the middle of the screen. When the player surpasses that border and moves while the camera is still, another variable of the player's movement needed to be changed to 'bounce' the player away from the house.



```
//determine from which side the player is coming from
if(playerWorldPosX < leftEdge && playerWorldPosY < topEdge && trueD <= maxD)
{
    console.log('COMING FROM TOP LEFT DIAGNOALLY');
    if(player.checkInsideCameraHorizontal() && player.checkInsideCameraVertical())
    {
        scrollPosX+=border;
        scrollPosY+=border;
    }
    else if(player.checkInsideCameraHorizontal() && !player.checkInsideCameraVertical())
    {
        scrollPosX+=border;
        player.ypos-=border;
    }
    else if(!player.checkInsideCameraHorizontal() && player.checkInsideCameraVertical())
    {
        player.xpos-=border;
        scrollPosY+=border;
    }
    else
    {
        player.xpos-=border;
        player.ypos-=border;
    }

}
```

```
else if(playerWorldPosX < leftEdge && playerWorldPosY > bottomEdge && trueD <= maxD)
{
    console.log('COMING FROM BOTTOM LEFT DIAGNOALLY');

    if(player.checkInsideCameraHorizontal() && player.checkInsideCameraVertical())
    {
        scrollPosX+=border;
        scrollPosY-=border;
    }
    else if(player.checkInsideCameraHorizontal() && !player.checkInsideCameraVertical())
    {
        scrollPosX+=border;
        player.ypos+=border;
    }
    else if(!player.checkInsideCameraHorizontal() && player.checkInsideCameraVertical())
    {
        player.xpos-=border;
        scrollPosY-=border;
    }
    else
    {
        player.xpos-=border;
        player.ypos+=border;
    }
}
else if(playerWorldPosX >= leftEdge-player.width/2-5 && playerWorldPosX <= leftEdge+player.width/2+5
&& playerWorldPosY > topEdge && playerWorldPosY < bottomEdge)
{
    console.log('COMING FROM LEFT');

    if(player.checkInsideCameraHorizontal())
    {
        scrollPosX+=border;
    }
    else
    {
        player.xpos-=border;
    }
}
```

Figure 15: House collision detection

Optimisation and user experience

A challenge evident from the start was making sure the users understood our concept. Thus, we made sure to include enough information, supporting the message and purpose of the project. Furthermore, we needed to improve the user experience and doing so by implementing various stylised icons and buttons, providing information and might not be as intuitive and thus making the gameplay clearer.

Lastly, there was optimisation. After finalising all elements, smooth gameplay was not possible – the character animations and movements were slowed down, and so were the sounds, making the whole experience unsatisfactory. Since removing any of the elements was not a viable option, as each served a distinct purpose, we needed to find a way to run the program with less computational power. Some of the measures we took were compiling near houses into bigger ones, so the array was shorter and thus the calculations, too, and reducing the number of particles in aura animations while keeping them comprehensive. Although these changes contributed to a smoother flow, it seems we have hit the border of p5.js's capability regarding computation and thus using a different library/environment to create the project might have been more efficient.

EVALUATION

Iterative process and skill development

During the development of our project we have learnt a lot about new libraries, new coding processes and the way in which we can tackle a problem. At the beginning of the project we thought it was better to develop every part of the game to the same level each time, this meant starting to make the interactions for the game stopping then going on to the video scene stopping, etc. This took a lot longer as we broke the flow of work. By deciding to split the work and setting targets for each other to hit. we worked out problems quicker, meaning we prioritised and got the sections of the game we wanted working correctly first. If we were to start another project now, we both believe that we have developed the skills to quickly decide what are the key points in the project and do them accordingly.

Learning new skills in development of graphics by using illustrator and p5.play to create more seamless animations, our game has improved from a rather basic design that we started out with to something more professional. By including computer vision in this project (which we had initially not envisioned) we created this game to be more realistic and interactive.

Improvements

Regarding improvements, we considered the option to mute the sounds. This seemed sensible if a user felt too uncomfortable or simply if they had hearing impairments and muting the sounds would make the program run smoother. However, we decided against it for the time being since it seemed to contradict some of our main concept and would thus not contribute to the experience but rather take from it.

Another addition to be implemented in the future could be the possibility of having multiple users playing simultaneously from their own computers. However, this might currently be too demanding for the p5.js library, a problem we've encountered before as abovementioned. Therefore, the last improvement could be choosing a different, more capable environment to write the program.

RESOURCES

Lundgren, E., 2014. tracking.js. [online] Available at: https://trackingjs.com/

McDonald, K., 2020. kylemcdonald/cv-examples. [online] Available at: https://github.com/kylemcdonald/cv-examples

ml5js·Friendly Machine Learning For The Web, 2018. [online] Available at: https://ml5js.org/

OpenCV, 2020. [online] Available at: https://opencv.org/

Pedercini, P., 2017. p5.play - a game library for p5.js. [Source code] Available at: https://github.com/molleindustria/p5.play/

References for images used for drawing the background: The Sprites Resource, 2020. [online] Available at: https://www.spriters-resource.com/game_boy_advance/pokemonfireredleafgreen/

References for the icons used in the game: Circle Bubble Icons, 2020. Icons8.com. [online] Available at: https://icons8.com/icons/bubbles

Veteanu, M., 2020. P5.Js Scene Manager. [Source Code] GitHub. Available at: https://github.com/mveteanu/p5.SceneManager/blob/master/lib/scenemanager.js