Interactive Audio/Video
Hannah Ödell, 266215
WS21/22

HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

FOCUS

Contents

FOCUS

FOCUS is a simple, game-like, audiovisual experience created for the module “Interactive Audio/Video” at the HFU. In Focus, the player uncovers sound sources that build up a soundscape. This is leaning into the topic of telepresence.
Through the direct interaction with the sound sources the player is more consciously immersed into the scene. FOCUS aims for the player to become more aware of the soundscape surrounding the player in real life, giving a closer sense of telepresence.

FOCUS was made using the free Godot game engine. Godot is a versatile game engine based on nodes that is relatively easy to work with.

Concept elements

List of scenes

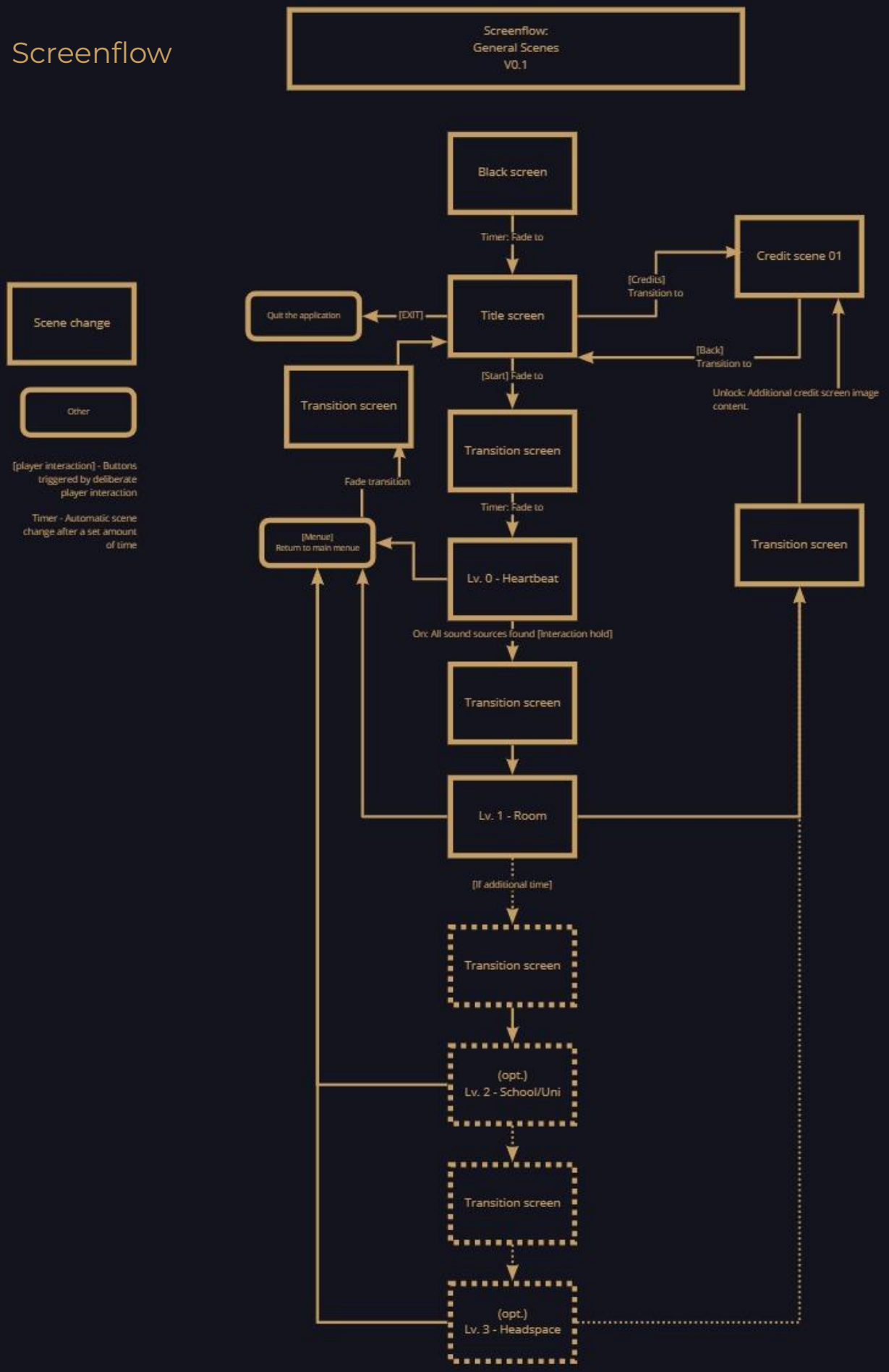
Main scenes/List of Levels

- Lv. 0 - Heartbeat (Tutorial/Waking up)
- Lv. 1 - Room (Realistic soundscape)
- (opt.) Lv. 2 - School (Negative soundscape)
- (opt.) Lv. 3 - Headspace (Hollow soundscape/Coping with overwhelming soundscape)
- Lv. End - Credits

Other scenes

- Title screen
- Transitions scenes:
  Text/Texture on background fading in/out
  (opt.) Animations in between

Screenflow


Screenflow:
General Scenes
V0.1
Scene change
Other
[player interaction] - Buttons triggered by deliberate player interaction
Timer - Automatic scene change after a set amount of time
Black screen
Timer: Fade to
Credit scene 01
[Credits] Transition to
Quit the application
[EXIT]
Title screen
[Back] Transition to
Unlock: Additional credit screen image content.
[Start] Fade to
Transition screen
Transition screen
Fade transition
Timer: Fade to
[Menue] Return to main menue
Lv. 0 - Heartbeat
Transition screen
On: All sound sources found [Interaction hold]
Transition screen
Lv. 1 - Room
[If additional time]
Transition screen
(opt.) Lv. 2 - School/Uni
Transition screen
(opt.) Lv. 3 - Headspace

The focus

As the player moves the player sprite around, it doubles as both the cursor and the means of detecting sound sources. For this document, rather than ‘player’ or ‘cursor’, we will refer to it as the *focus*.

The focus itself is animated, and has multiple states it seamlessly can transition to. Initially, the focus was planned out like this:

State 00 - Spawn

Spawn/despawn animation - Fade in

State 01 - Idle 01

Closed eye

The focus doesn’t detect anything or is currently inactive/hasn’t been active for five seconds.
A default sound is played on hitting [interact], a barely visible, differently colored ripple is displayed, and - if not stated otherwise for the scene - the eye will open.

State 02 - Idle 02

Open eye

The focus doesn’t detect anything.
Occasionally blinks.
A default sound is played on hitting [interact], and barely visible ripples are displayed.

State 03 - Detect Far

Open eye with small circle

The focus detects something, but is far away from it.

An obscured version of the sound is played on hitting [interact], and more clearly visible ripples are displayed.

State 04 - Detect Close

Open eye with intricate pattern

The focus detects a sound source close by.
An obscured version of the sound is played on hitting [interact], and fully visible ripples are displayed.

State 05 - Clickable

Open eye with dot in the centre

The focus detects a sound source directly underneath it/within interaction range. A dot appears, indicating intractability, and the cursor emits a differently colored ripple which stays.

(opt.) A particle system emits particles for further player feedback. Flickering glow above the dot?
On pressing and holding [interact], see state 06.

State 06 - Concentrating

Open eye with abstract shapes concentrating into its centre

The animation of State 06 gets initiated by holding [interact] while in stage 05 (A little bit of 'latency' to the animation, to make it feel heavier). A concentration sound will begin to play.
The ripple will be 'sucked in', then State 06 will play the beginning of its animation.

(opt.) On releasing [interact] too early, the animation will play backwards from where it left off. Otherwise, it will proceed to State 07, then back to State 01.

State 07 - Cooldown

Open eye, center fill fading out, creating a seamless transition back to State 01 or State 02.

(opt.) The screen flashes shortly in a flash of light.

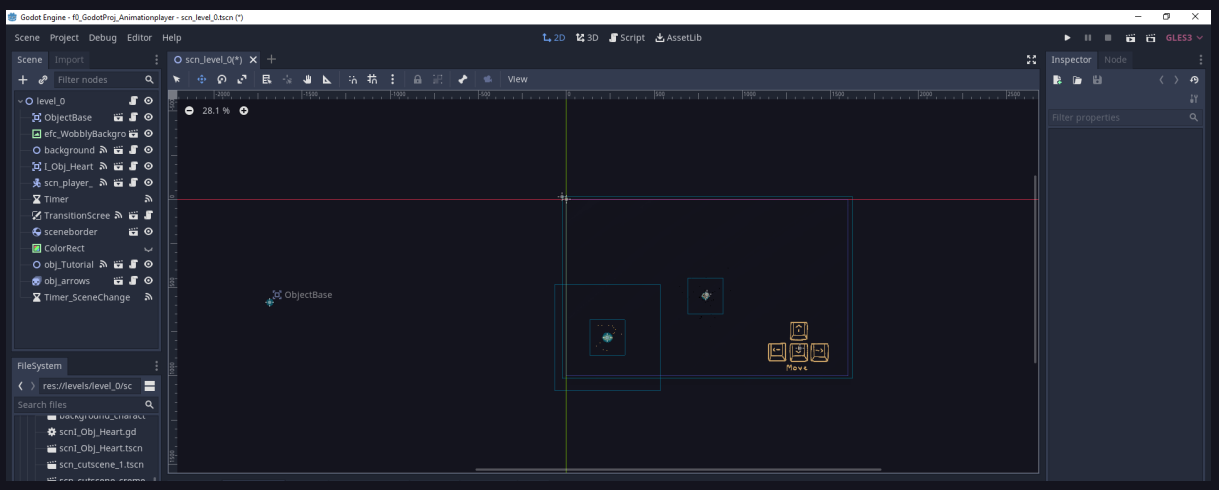
Nothing happens on hitting [interact] for six seconds, and the focus fades out for a few moments while the sound source fades in.
(Quick fade out, slow fade in)
After six seconds of cooldown (in which the player can still move, but not interact), the focus will very slowly fade back in.
No ripples will be displayed on [interact].

After playtesting and uncovering multiple issues with the player, the ripple mechanic was shown to be excessive without real use to the player. State 04 and State 05 were combined into a single state, and instead of three iterations of player animation towards the findable object, it got reduced to two.

The player functionality was one of the main hurdles for the project. With little to no programming experience the most simple things became arduous tasks. To have the focus easily instanced into any level or scene, most vital functionality rested in the focus code, resulting in a very long piece of script on the player, a lot of signals connected to it. During the creation process, the focus was and still is the one acting up most, throwing errors at every possible moment.

Needless to say - The player object/the focus is one of the reasons why the project ran into so many hurdles until I asked a more experienced friend for help. In theory, and for someone with coding experience most likely in practice, having a moveable node that interacts with objects and changes his shape according to distance is easy.
In practice, not fully understanding the tree hierarchies, tools and possibilities in godot lead to all sorts of trouble with that concept.

The focus was redesigned, remade, set up newly in instance over instance. Learning while coding also meant returning and cleaning up terrible coding practises regularly - For example: An old version of the player distance detection, for playing the correct animation at the correct distance, consisted of three match-cases within each other.
Movement had been located on the focus first, then on the level scene, then on the focus again - instancing, extending scripts and calling external scripts is incredibly easy to do in godot, but it took a while to notice these are options at all.
One of Godot's most useful tools are signals.
Signals can be emitted in any script easily. They travel up the node tree and can be received by other nodes, telling them what to do. As long as all nodes are within the same scene, already instanced, they can be manually connected. But the focus needs another object in the same scene with it to function and not throw an error at all times - Specifically an object from the class ObjectBase. Even though the heart, the furniture and the buttons are all derived from the ObjectBase, there needs to be the specific ObjectBase present on the scene or it crashes and the signals can't connect.
As such, every level in the application has an ObjectBase in it, pushed far off screen.



The goal was that - With the base code set up, anyone can easily make and drag in any sprites with sound.
It is free to everyone to use the assets and the code, adjust things as long as they are kept under their original copyright (Attribution licence). The playable application was exported and uploaded on itch.io, but the original code can be downloaded, tested and cloned in the github repository.
To play the application on the computer, clone the git repository, download godot, and import the project.godot file in the launcher. Hitting the Play button in the upper right corner will launch the application.

Itch.io: https://pr0crastigam3s.itch.io/f-o-c-u-s
Github: https://github.com/Hanhan139/F-O-C-U-S---Interactive-Audio-Video

Previous ideas

The concept of FOCUS went through multiple stages. Initially, finding an object was to be tied to a rhythm based mini-game. However, the realisation of rhythm games turned out to be very difficult, with few resources and tutorials available for a beginner. Another idea was to turn it into a generative, touhou-style bullet hell game, where finding an object would cause more bullets and bullet patterns to spawn until all objects in a scene have been found.

Ultimately, both ideas were too elaborate and I had to adjust my expectations, since I was too inexperienced at programming, but it would still be a fun idea to work on those aspects for the future.

The original concept also included multiple Levels, as mentioned above.
The tutorial area should demonstrate the mechanics to the player with one example object. The first level was the goal I set myself to create, as it exemplifies the building of a soundscape.
The following levels centred around the subject of sensory overload through noise, touching on a more personal topic to me. Sensory overload is very common with both people on the autism spectrum, as well as in people with ADHD. Distracting sounds do not allow a person to focus on anything else, and it is extremely hard to get back to the task at hand. Being overwhelmed by sound can in part be perceived as being physically painful, and causes outbursts of aggression, frustrated crying or impulse reactions. This would have been something I would have liked to explore in level 02, where the player tries to focus on something that is said in the top of the screen, but the focus gets teleported to and reveals another sound source each time. Chattering, whispering, smacking, clicking of pens, and the clicking of clocks that had been portrayed as comfortable in the first scene.
On the other hand, peculiar sounds can help to calm down, which I would have liked to explore with an abstract scene in level 03. Level 03 plays in the character's head. The scene is filled with the sound of repeated, dull impacts, as the character is bashing their head against the wall.
Head banging and other self harmful behaviour is also very common, and easily caused by sensory overload - In such cases I found specific melodies and noises to be soothing, and often look for a specific song or sound. Level 03 would consist of such sounds.

Level 04 was originally intended as an ending, getting outside and listening to music giving a sense of closure to the game.

3. Development process for Interactive Audio/Video

First I was trying to get to know Godot with the help of a friend. We entered the WGJ (Weekly Game Jam) to understand the Engine a little more, and made the little Kettle game.
https://pr0crastigam3s.itch.io/wgj196-strangerecipe-kettle
I watched a variety of tutorials and tried to recreate them in the process.

Concerning Audio, I looked into a variety of tools for FOCUS.

I started experimenting with JummBox, an online tool to create chiptune melodies. It saves all information within the URL and is rather intuitive. A few examples can be found in the GIT folder including Links to the JummBox version.
FOCUS_Game\ConceptSketchesAndOther\JummBoxExperiments

Here is one example:
https://jummbus.bitbucket.io/#j3N0hObfuscous%20Gamesn820s0k0l00e0dt1Um0a7g0dj08i0r1O_U0000000000o3212222200T3v0tL0TaD0Ou03q1d5f8y2z7C0SU0M51pr2qiqqrrrc0h0T0v0pL0PaD0Ou00q3d4f7y1z9C0w0c0h3T1v0bL0OaD0Oub3q1d2f7y2z9C0c0h0A1F0B5V4Q00acPd559E0119T1v0dL0OaD0Ou01q1d7f4y3zkC0c0h0A3F6B0VeQ7337Pd1a8E02dgT6v0pL0NaD0Ou06q3d3f8y0z2C1c1WC1h0T0v0fL0TaD0Ou14q1d6f9y2z1C0w5c1h2T5v0bL0EaD0Oue8q3d6f7y1z8C0c0h6H-SstrsrBzjAqjhT1v0iL0OaD0Ou01q1d6f7y0z6C1c0h0A5F2B6V7Q0530Pf636E0011T3v0iL0HaD0Ou03q1d5f9y1zfC0Sh990999irABZBSTc4h0T3v0hL0WaD0Ou03q1d5f7y2z6C1SLXtHrrsksrzrrrrc2h0b4x8i4QlDC0018yc0hmsh4g000000000018zhg000000084zg0014x03hD4yd4gBo0000i018Q00oh4i8z014h4h40000h4gp2feIPnQl9P2PuhrrJCSrijrtlSOPbZKFB-9bCq-0o4APnQkksKqIyyz0q2pjwkNth7ibE6yBczMoKEAtoKXFIYakjbZ2p7EEFCLFLjatqoKYzF5TpUKIr7BeNUKaJOP8ZvFAuD8Bj8ZzQQCpuxChV5Fj8ZeheiCnXZjXqVRpeh9MtJp6j97w1BSpHUJqU3mpjwppyX98WF5Q3JcRSP8Y78KEOhSObLZFBHAuNV7A6Z8zjatioKAzF5tdnd70s3-KXLA5HqqKkXt2jbZADb5cnthhgqZ0J0I5hhqsvpyWIj5N3jiP8Zdap7IuDQPbQqyQ9AuD8D9jbYBGvrg2ubH6NVjIubyHsIOfnWp7FO9kOfoZd9CnEpAuhqkOfjAjAFB-_k-SK9b4ugs70As0s7PF8YwUeEzFcKyCCkV5cn9AuPqbB4ujmbC4uiCq-AbcOYaMIOf9PCcCne2bVsIPnMz9S9SpTdMehmoKhAzIAnt9v4PMgRBACnm2c6ne_SCr6e3pydcngbgbgbf5RzoYFSf5Nq2MSp7C9xj8ZehaChX7FFcOZ3czObiChWsysBcLOmFZJ9Vvn896ChXyewUewzwjnAt97ghQCn04t170BYzw3bw3bIcKMLAt1BS6ngnRCmAthBR6nmpthv8W3bGbWOe18UczG8OPdvIADcbdF5JKSrpJ9dmQWCq-BoSfatzNsNmMmp7EoOeCidcRZjczRJ17j96CnV74Zj8ZlcWvstfefjefRCKap7yqLHwi-wzAGyeyQmEnonRS4qoKEn0ieAzGHcL0CnshR7pX3bG8VcXfknVoJFUzwhrOewzGd71ch8YMlcLQ5UFAogw2ZcknhhqbjjhBlpEJgJt550Jtd7aGONqxqWaaaGSoEGYDpGONqxqWaaaGZGYDIRpoJgJt555luRujIRpoJgKhlJkbhgE5ckthhhqbibhhhhibibhhhgChzsAmyyyyGxCa8D900

After that, I tried editing those files in Audacity. Adding effects, looping the audio, and downloading additional sounds from freesound.org to add to the song.

https://youtu.be/ASfGmrNIFRs

But I wasn't sure if the chiptune sound would necessarily fit what I was going for, so I also tried working with LMMS, downloading plugins and using them.
I found a beautiful piano plugin, which was then used to create the title theme for the application.
https://youtu.be/pPojv_mJhCY

LMMS was also used to create the piano chimes and sound effects for finding an object or interacting with the UI.

The object sounds are taken from https://freesound.org/ and edited in Audacity. There is one version of the sound that plays for the soundscape, and one which is distorted when entering an area close to the findable object.
Godot already allows for spatial sound in a 2D scene. The AudioStreamPlayer node plays a sound globally (Piano chimes), and the AudioStreamPlayer2D adjusts the audio output spatially (Object sounds). But for some of the soundscape, f.ex.tThe cars out the window alongside the wind and bird sounds, I duplicated the sound and faded it to sound like there is directionality to the travelling cars.

The video/visual parts of FOCUS were made using Clip Studio Paint Ex. Clip is a drawing tool with particular focus on anime, comics and cartoons. It supports hand drawn frame-by-frame animation, which makes up the bulk of the assets and animations in the application. Everything from the focus, to the objects, to the character has been digitally hand animated.

Animation for game assets comes with unique challenges. Most assets need to be infinitely loopable. I had to keep in mind in which state to draw and animate an item/start the animation, so that it fits to the sound. Especially the clock posed a problem in timing the ticking, but the issue was later solved with a Timer that repeatedly calls upon itself.

Exporting most animations in white allowed to modulate the color later in Godot, in case it needed any adjustments. The exception to that is the focus. Since it has a white core, coloring it with modulate would also color its core, so any adjustments to the player had to be done in Clip Studio.

Outlook

My experience with FOCUS was a very two sided one.

The project allowed me to look into a high variety of different programs.
Godot specifically is an incredibly powerful tool - it is open source and surprisingly easy to get into. However for a beginner, it sometimes is hard to determine exactly why something isn't working.

Working myself into so many programs allowed me to see how many aspects combine in a finished project. This is potentially useful when delegating tasks in future projects and estimating the workload. But since there was little time spent with many new programs, the result could have been better. Especially the fact that the player took so long to figure out to this extent frustrates me. And the player still has issues if the spacebar isn't held, put simply pressed.

On the other hand, this is valuable experience in many topics that I wouldn't have tried out without this module. Especially looking into audio workstations, since they are immensely intimidating.
As for Godot, with the experience I have collected over the course, I can actually say I enjoy working with this game engine.
Many of the previous ideas could be implemented in future iterations, and I can wholeheartedly suggest Godot for 2D Projects to other participants.

All audio sources and assets are listed and available for use in the git folder, in case it might be of interest for a future project.