RE-EXPLORING VIDEOGAMES FOR AUDIOVISUAL PERFORMANCE AND COMPOSITION

berk özdemir -2020 - artscience - master's thesis

This text aims to give an introduction on how videogames can be approached beyond gaming and used as a raw material in creative artistic practice, by theorising game elements such as sound, space, gameplay and form structure in musical composition and theory terms.

What is Inside:

INTRO _____ 1

THINKING - WORKING WITH SOUNDS _____ 4

Sound Palettes _____ 4

- The genre, context, narrative and art style
- Technology / medium / hardware

What/who Plays Which Sound? _____ 6

- User input
- Objects, NPCs (non-player characters), other players
- Spawns / alerts
- Non dynamic background sounds, soundscapes

THINKING / WORKING WITH FORM _____ 9

- Musical Form == Game Form _____ 9
- Speedrunning _____ 9
- Form Analysis - Labeling _____ 10
- Questions _____ 11

EXERCISES ____ 12

1- Replace the Audio Files of a Videogame ____ 12

Example: SERIOUS SOUND RANDOMIZER ____ 14

2- Be a Wanderer in an Open World Game ____ 15

Example: Skyrim Recordings ____ 16

3 - Compositions ____ 17

Examples:

Etudes for Fighting Games ____ 17

PPPB (Pseudo Player Performer Bots) ____ 19

OUTRO ____ 28

BONUS ____ 29

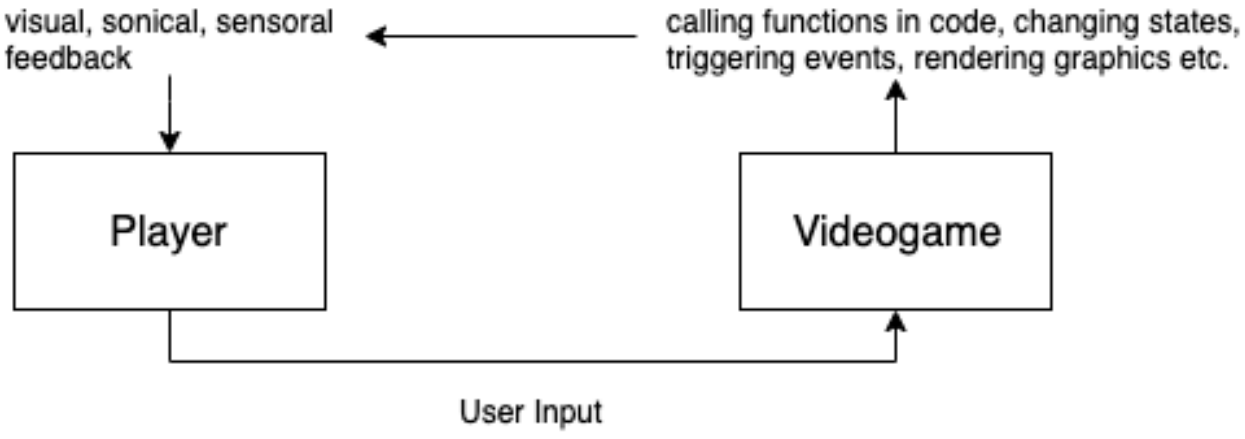
ALL LINKS ____ 30

INTRO

We are currently living in an era where we are getting bombarded by new videogame releases on different platforms everyday. The amount of people enjoying playing videogames, from any age and any type, are increasing exponentially day by day, with the evolution of new ideas, technologies, end user devices, game elements, platforms, interfaces, mechanics, game engines, multiplayer capabilities; thus the videogame industry dominates the entertainment industry, surpassing Hollywood and music industry in the market value and net worth; impacting the culture of generations, and keeps growing.

I have been involved with video games of different kinds since I was a child. They have become a major inspiration to me to like many people in my generation; I've explored and became a part of new worlds, stories, characters in single player games; played offline and online with people, and had a great time with them working on the objectives together; or competing with each other.

As a consumer of these games; there is one thing that is bugging me a lot for a while: we gamers / consumers / users are spending hours / days / weeks on these complex pieces of softwares, which can be considered as Gesamtkunstwerk - "combination of many different media and mechanisms in a single piece of work", and we usually only expect a single dimensional experience and interaction with those giant virtual worlds and narratives the developers designed for us, as the way they want us to do. But why do we have to only "play" the games? There is already a vast library of videogames laying under our feet, which still carries infinite number of hidden potential use cases we still have not discovered yet.



What I will propose in this text is an alternative look at ready-made videogames. Videogames are complex piece of software that continuously run lots of code in real time while we are playing them. Our actions are sent from an interface (mouse, keyboard, gamepad, joystick, HID device, sensors / trackers) to the computer, console, or a server, calling some functions and events to change the states and events in the game. Then we perceive the feedback of our actions mostly through our eyes (displays, projection, googles), ears (headphones, speakers), or our different organs (gamepad - cellphone vibration).

To put it more simply;

- Videogames are playable sound and visual generating softwares.
- Gameplay controls are instrument interfaces.
- Game space is a interactive performative space.
- Players are performers. Even in some cases, NPCs (non-player characters) are also performers.

Each reader of this text will approach the games differently; depending on their interests, professions and disciplines. I will be sharing my approach on specific elements of games I worked with, such as sound, space, gameplay; how I turn them to a new playground, as raw-material for creativity and exploration, getting help from some methods and concepts from musical composition practice and theory. After the theory part, there are three exercises (alongside my experience with them); to encourage the reader to start with experimenting.

THINKING – WORKING WITH SOUNDS

"Find a video of a game footage with no commentary and music, then listen to it with eyes closed. What do you hear?"

Sound is one of the fundamental elements in videogames, and plays an important role as a companion to visuals, to give a fully immersive experience to players.

It helps the player to make sense of game world they are in, gives feedback of their actions, gives clues about the changes happening around, even if they can not be seen.

A sound in the videogame does not have to be present all the time (with the exception of background music, or sounds such as environmental soundscapes). Sound events are triggered at specific moments, as a function call in the game, with a determined duration.

In this chapter, we will be looking at how sounds are chosen for a videogame, and what / who may trigger which kind of sounds.

Each game comes with its own set of sounds (sound palette), depending on:

1 – The genre, context, narrative and art style:

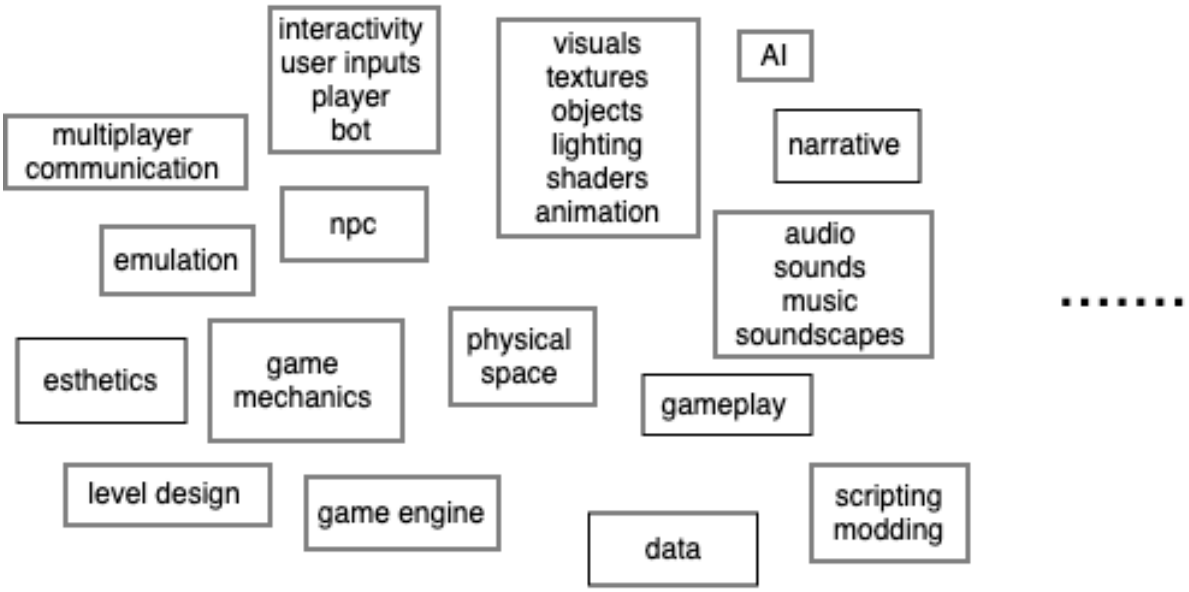
The types of sounds heard in a game are chosen to serve to represent the concept and depiction of the environment; so they should be relatable to the game world and experience. Different stylistic choices for the same game, would result completely different experiences and feelings. One could experience a goofy game instead of realistic, stimulating instead of relaxing, if the sound material was different than original.

For example, a basketball game simulation represents a basketball match. Then the sounds a player hear while playing a basketball game could be:

- **Ball bouncing, hitting hoop**
- **Basketball players running around**
- **Cheering crowd**
- **Commentator**
- **Announcement alerts (quarter end, fault, pause)**

CHOOSE A VIDEOGAME

STUDY / EXPERIMENT ELEMENT(S) OF IT



DO WORKS WITH THEM

automation
sonification
communication with other software
audio/video recording
screenshots
fine art
performance on stage
performance on couch
generative pieces
new narrative
new gameplay
new concepts

.....

Or in a first person shooter game, the main sounds could be:

- Gun shots, item usage (med packs, boosters)
- Item pick up
- Enemy noise (human, monster, alien, animal, robot)
- Movement (walking, jumping), which can vary by what kind of material the player is stepping on (grass, metal, glass)
- Objects reacting to movement or weapons (glass break, wall explosion, falling objects from the shelf, collision of cars)
- Alerts (on low health, low ammo)

Open world fantasy role playing game in medieval era:

- Soundscape and music of the environment and era (farm, tavern, castle, landscape)
- Items, weapons that are picked or used
- NPCs (humans, monsters, animals)
- Alerts of skill cooldowns, leveling up, change in a player status (hunger)

Real time strategy game:

- Units in movement or in action (workers, builders, attackers, defenders)
- Buildings that are created (hubs, houses, fortresses)
- Alerts (units are attacked or dead, the construction is done)

2 - Technology / medium / hardware:

Every technology, medium or hardware a videogame is built on, comes with its own freedom and limitation. Back in 80-90s, the sounds were usually generated by dedicated sound hardwares (mostly sound chips), which were far from being capable to play realistic sounds; but this limitation brought its own unique aesthetics, and composers, sound designers and programmers found many creative ways to depict the game elements, and make music.

As for today, there are lots of options a developer can choose from, to use any kind of sounds they want. They can use sample playback, procedural sound processing, generative models, or any other synthesis and digital signal processing method. Typically a contemporary video game consists of sounds being generated by any combination of these methods.

WHAT/WHO MAKES WHICH SOUND?

While we are playing a videogame, it is highly possible that we might hear many different kinds of sounds at the same time. To distinguish between the types of these sounds, I will examine them under 4 different categories based on what / who produce them.

1 - User input:

This category contains the sounds that are made by the player. The interaction between player and game is made through an interface (keyboard, mouse, gamepad, joystick), with a set of inputs (button presses, mouse clicks, joystick movements), mapped to control and interact with the characters, units, game environment, menus, and other controls. Each user input (or the combination of inputs) sent to the game, consequences a single event or chain of events. Game code reacts to the user inputs as changes in the state of game, and shows the reflection / feedback of these changes as image and sound to the player.

Examples:

> A fighting game:

Player presses a button, which is mapped to punch → Punch sound

Punch hits the enemy → Hit sound

> A first person shooter game:

Player has a grenade as the active weapon.

Player holds button x, character releases the pin of the grenade → pin release sound

Player releases button x → character throws the grenade → bomb bounces on walls and ground

After 3 seconds → Bomb explosion sound

> A platformer game:

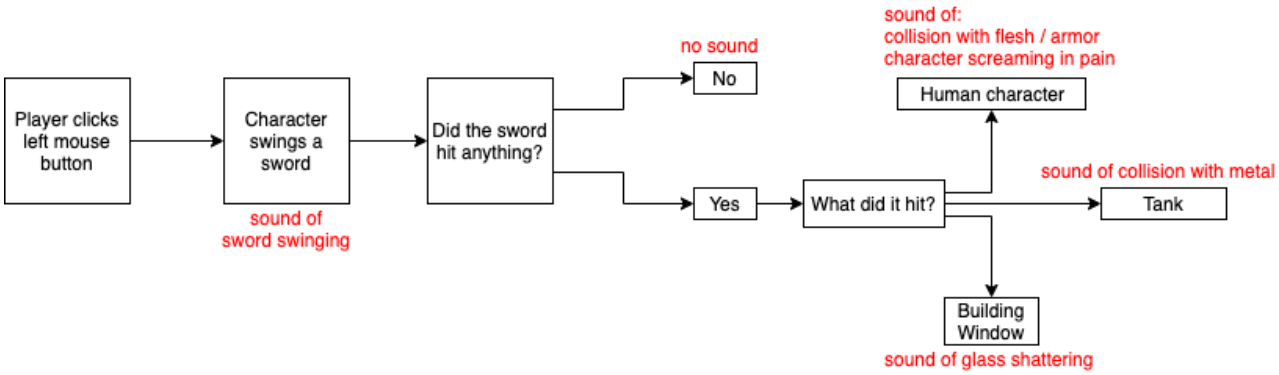
Player presses "x".

Character jumps → Jump sound

Players lands on ground → foot hitting the ground sound

Player falls → you are dead sound

> A visual example of an action / adventure game:



2- Objects, NPCs (non-player characters), Other Players:

These sounds are produced by the entities in game which player has no direct control.

Which may include:

Non-living objects, which make noise by themselves continuously (like a ventilator running), or make sounds when a player / NPC interacts with them (like a glass, vase, wooden plank).

NPCs (non-player characters), which move by themselves and react to the player, depending of what kind of behavior is coded for them (which can be predetermined, algorithmic or responsive).

Other players: in a multiplayer game, the other players usually produce the same sounds that the player produces. But they are controlled by other people.

3- Spawns / Alerts:

These sounds are played by the game in a specific locations and times, to give information to the player of the state / change throughout the gameplay.

e.g:

- **The player is below n percentage of health, resources.**
- **New enemy spawned.**
- **An NPC noticed the player.**
- **New information is obtained**

4- Non-Dynamic Background Sounds, Soundscapes:

Music and environmental soundscapes can be considered an example of this category. Compared to other categories, the sounds in this category usually are statically played as continuous loops on the background; which means that there is no change, or very subtle change over time on what a player is going to hear, unless the player goes to another environment.

THINKING / WORKING WITH FORM

Form in music, is the organization and structure of sound events in a composition or performance.

A musical piece may consist of many different musical elements; sections, phrases, melodies, chords, motifs, rhythms, sounds, text, movement, light... And the form is like a melting pot, which builds a structure and narrative out of them, by distributing these elements in a time structure, to make the music stand on its own as a piece.

Videogames are similar to music in means of form structure. The game elements, such as story, level, entities, objectives, visuals, sounds are structured to determine how long and how a player will spend their time with the game, how many different things they will encounter and do, and in which order they will see them.

Deconstructing the form structure of a music piece reveals a lot of information on how composer wants the audience to perceive the musical elements they introduced in the music, their roles, and relation with the other elements. The same thing also applies for videogames. If a player gives their attention to the game elements they encounter, and to the changes, repetitions, transitions between them, they can find out how the game form is structured, and how they are expected to spend their time by playing that videogame.

Speedrunners:

Before going to analysis of form in videogames, I want to give a quick shoutout to speedrunners, who inspired me a lot to write this part.

Speedrunning is a type of playthrough, where player aims to complete a level or the whole game as fast as possible.[1] Almost every videogame has an active speedrunning community, determined to find new shortcuts, skips, glitches to beat the game faster then before.

I see a connection between speedrunning practice and game form analysis methods, because speedrunners have to do a lot of research on game mechanics, elements, code to understand how the game is structured, and they bend the gameplay in ways that developers would have never thought.

And a speedrunner usually has to practice a lot to memorize and master all the tricks, patterns, button combinations, like they are practicing a piano to learn a virtuosic classical music piece.

Watch Half-Life 2 Developers React to 50 Minute Speedrun:

https://www.youtube.com/watch?v=sK_PdwL5Y8g

1 https://en.wikipedia.org/wiki/Speedrun

Labeling:

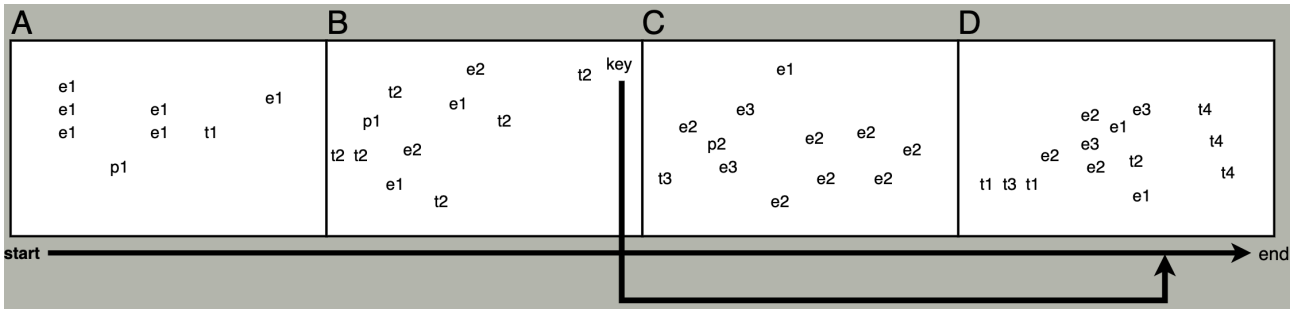
For describing form of a music; one of the most useful ways of classifying and distinguishing different musical elements (in different scales) is labeling them with letters, numbers and symbols.

To help the reader about labeling structures of a music, I would recommend them to check this Wikipedia page: https://en.wikipedia.org/wiki/Musical_form#Labeling_procedures

Labeling also works pretty well for analyzing form structure of videogames. I want to show an example on using labels to describe the form structure of an imaginary videogame level.

So, let's assume that we are playing a level in a 2D platform videogame, which has almost identical game mechanics to the first Super Mario Bros (1985)[2].

In this game, the main goal of each level is simple; the character starts at the leftmost point of the map, and it needs to go to the end of the level (which is the rightmost side), by avoiding traps and enemies along the way. The game only lets the player to go right; so the player can not return to a point they passed.



There are 4 types of entities in this level in total; some of them are beneficious (key, power-up), or malicious (enemy, trap). I will label them with string+number combinations or a word.

- 3 different types of enemies, labeled as e1, e2, e3
- 2 different types of power-up items, labeled as p1, p2
- 4 different types of traps, labeled as t1, t2, t3, t4
- 1 secret key, labeled as key

The level consists of 4 different sections. Each section is unique, with a different set of entity placement. When the player starts at A, there is only entities e1, p1, t1. As player arrives to point B, new enemies and traps start to appear (t2, e2).

[2] https://www.youtube.com/watch?v=1qcTwuKozs8

At the end of B, there is a secret door; which teleports the player to the end of the level (saving the player from having a lot of trouble in C and D). This secret option gives the player to finish the level with A → B → D route.

At C, we see a horde of enemies (e1, e2, e3), a new power-up (p2), and a new trap (t3).

D is all the enemies and traps that has been shown before, and close to end, there is a new trap kind (t4); which are placed right before the ending point, to surprise and challenge the player.

Questions:

I would suggest the reader to ask questions related with form structure of a game, while playing it, or after the gameplay session. I wrote down below a couple of example questions to start with:

- How many different sections did I encounter in the game? If there is only one section, do I see and do the same things every time I play, or are there changes or variations?
- If there are multiple sections, what is the relationship between them? Are they part of a same narrative, or irrelevant to each other? How are they connected? Can I move between these sections freely? Or does the game force me to progress linearly?
- How much space does game provide me for exploration?
- What are the repetitions?
- What are the changes?
- How would I label the game elements I've encountered?
- What is the proportion of gameplay / cutscenes? (Interactivity, non-interactivity)

EXERCISES

In this section, I will be sharing 3 exercises the reader can take as a starting point for experimenting with videogames.

1 - Replace the audio files of a videogame:

In this exercise, we will be focusing on video games which plays audio samples for generating sounds, rather than the ones that use procedural synthesis or other methods.

An audio file can be associated with any event that triggers it; e.g. soundscape of game space, user inputs, item usage, interacted object, moving things, materials.

Depending of the complexity of the game, we might hear a couple, dozens, or even hundreds of different sound files being played at the moment.

So even replacing a couple of sound files in the game data:

- changes the whole gameplay experience, which provides replayability of that game in new contexts and means. There are already many examples of people using this method to create fun video content, which can be found by searching in Youtube with keywords "replaced all sounds with"[3].
- gives tips on how the sound engine / mechanism works in the underlying game code; it gets easier to understand how/when/why the sounds are played.
- reveals a hidden functionality of the game that was actually waiting for us to use: a powerful instrument, totally abstracted from visual elements we see; which provides a new space for experimenting with musical concepts, such as timbre, rhythm, harmony, form, orchestration.

To play with the sound files of a game, I would suggest the reader to take these steps:

1 - Extract the original audio files to a folder; analyze their properties:

- duration ⇒ short, long
- amplitude ⇒ how strong is the sound compared to other ones?
- loop ⇒ does it play once per event, or loop?
- effects / modulation ⇒ reverb, delay etc.
- panning, distribution in space ⇒ mono, stereo, binaural, dolby

[3] https://www.youtube.com/results?search_query=replaced+all+sounds+in+with

- envelope ⇒ change of amplitude / filter / modulation over time.
- any randomized parameter? such as playback speed,
- characteristic ⇒ one shot (door open), loop, drone / ambient (rain, wind, machinery)

2 - Collect a combination of audio files to replace the original sounds in the game. Which could be:

- different notes of a single sound source, on a musical scale of choice.
  e.g. sound sources: piano, sine wave, xylophone, flute ...
  e.g. scales: dorian, aeolian, major pentatonic, microtonal scales, random frequencies ...

- a single audio file; with
  fixed or variable durations,
  fixed or variable amplitudes,
  fixed or variable attack / release times

- randomly or algorithmically generated audio files
  random audio cuts from Youtube, Freesound, radio, your own audio files etc.
  an algorithm which generates audio files with randomized parameters

- pronunciation of letters - syllables of a language

- audio recordings of the sounds of your home

- audio files of "x" game, but reversed

3 - Use your sound pack / samples to replace

- all files
- a single file
- randomly selected files
- only sounds of a spesific class (e.g. enemies, soundscapes, character dialogues, gunshots, doors)

4 - Instrumentation

- Use different combinations of sound packs for different classes of sounds.

5 - Play with your new sound packs, record your experimentations, listen to them.

Example: SERIOUS SOUND RANDOMIZER

To do this exercise, I had to do a research on which games in my game library would work best for my needs, without going in too deep with hacking or reverse engineering the game code.

Every videogame is designed differently, so it means that each game has their own ways of storing / exposing game files. Unfortunately, many of the videogames use their own special file formats; and contains the files in a type of big archive file; which makes them hard or impossible to reach and replace.

(Serious Sam[4] Logo)

```
'Enemies/UghZan/Laser.wav',
'Enemies/UghZan/LavaBomb.wav',
'Enemies/UghZan/Punch.wav',
'Enemies/UghZan/RocketLauncher.wav',
'Enemies/UghZan/Smash.wav',
'Enemies/UghZan/WalkL.wav',
'Enemies/UghZan/WalkR.wav',
'Enemies/UghZan/Wound.wav',
'Enemies/WereBull/Death.wav',
'Enemies/WereBull/Idle.wav',
'Enemies/WereBull/Impact.wav',
'Enemies/WereBull/KickHorn.wav',
'Enemies/WereBull/Run.wav',
'Enemies/WereBull/Sight.wav',
'Enemies/Zorg/Death.wav',
'Enemies/Zorg/Fire.wav',
'Enemies/Zorg/Idle.wav',
'Enemies/Zorg/Sight.wav',
'Enemies/Zorg/Wound.wav',
'Enemies/Zumbul/Death.wav',
'Enemies/Zumbul/Fire.wav',
'Enemies/Zumbul/Idle.wav',
'Enemies/Zumbul/Sight.wav',
'Enemies/Zumbul/Wound.wav',
'Environment/Laser.wav',
'Environment/rain.wav',
'Environment/rain_02.wav',
'Environment/wind.wav',
'Environment/Thunders/Thunder2.wav',
'Environment/Thunders/Thunder1.wav',
'Environment/Water/River.wav',
'Environment/Water/Waterfall02.wav',
'Environment/Water/WaterFallLarge.wav',
'Explosions/Explosion01.wav',
'Explosions/Explosion02.wav',
'Explosions/Explosion05.wav',
'Explosions/ExplosionHuge01.wav',
'Explosions/Splat01.wav',
'Explosions/Splat02.wav',
'Explosions/Splat03.wav',
'Fire/Burning.wav',
'Fire/FireTorch_02.wav',
'Impacts/Acid/Acid_Debris01.wav',
'Impacts/Metal/Metal_Crash_01.wav',
'Impacts/Metal/Metal_Debris01.wav',
'Impacts/Plasma/Impact_01.wav',
'Impacts/Stone/Impact_Stone01.wav',
'Impacts/Stone/Impact_Stone02.wav',
'Impacts/Stone/Impact_StoneLarge01.wav',
'Impacts/Wood/Impact_Wood_Large01.wav',
'Impacts/Wood/TreeBreak.wav',
'Impacts/Wood/WoodDebrisFall.wav',
'Items/Key.wav',
'Items/Weapon.wav',
'Items/Ammo/Ammo.wav',
'Items/Armor/ArmourMedium.wav',
'Items/Armor/ArmourShard.wav',
'Items/Armor/ArmourSmall.wav',
'Items/Armor/ArmourStrong.wav',
'Items/Armor/ArmourSuper.wav',
'Items/Armor/Generic.wav',
'Items/Health/Generic.wav',
'Items/Health/HealthLarge.wav',
'Items/Health/HealthMedium.wav',
'Items/Health/HealthSmall.wav',
'Items/Health/HealthSuper.wav',
'Items/Powerup/ExtraLife.wav',
'Items/Powerup/Jump.wav',
'Items/Powerup/Powerup.wav',
'Items/Powerup/Serious_Damage04.wav',
'Items/Powerup/Serious_Jump03.wav',
'Items/Powerup/Serious_Score03.wav',
'Items/Powerup/Serious_Strength03.wav',
'Items/Treasure/Generic.wav',
'Items/WeaponPickup/WeaponPickup01.wav',
'Materials/Bullets/Bullet_Acid01.wav',
'Materials/Bullets/Bullet_Default01.wav',
'Materials/Bullets/Bullet_Feather01.wav',
'Materials/Bullets/Bullet_Fire01.wav',
'Materials/Bullets/Bullet_Flesh01.wav',
'Materials/Bullets/Bullet_Grass01.wav',
'Materials/Bullets/Bullet_Ice01.wav',
'Materials/Bullets/Bullet_Metal01.wav',
'Materials/Bullets/Bullet_Stone01.wav',
'Materials/Bullets/Bullet_Stone07.wav',
'Materials/Bullets/Bullet_Water01.wav',
'Materials/Bullets/Bullet_Wood01.wav',
```

Also, modifying game files are usually considered as an illegal activity (especially in multiplayer games); thus game developers prefer to design the game file system such a way to make them inaccessable.

On the other hand, if a videogame is moddable, it is highly possible that their audio files can be accessed and replaced easily. Serious Sam[5] is one of them.

Serious Sam has been one of my favourite videogame series since my childhood. Their game genre is FPS, and each game is based on this same formula; the protagonist Sam enters to lots of different places filled with hordes of enemies, and he has to progress through levels by killing all monsters he encounters, and solving puzzles, getting help from the new weapons, powerups he collects.

There are around 300 audio files in Serious Sam HD: The First/

[4] https://en.wikipedia.org/wiki/Serious_Sam

Second Encounter[6]'s game directory, which covers almost all the sounds being played in the game (with the exception of a couple of dialogue and weapon files), stored in different folders for each sound category. Since it would be too time consuming to rename all the files for replacement, I wrote a short script in Python, to automate renaming / replacing all audio files randomly.

The reader can use this link to access the source code (with instructions):

https://gist.github.com/vortextemporum/18a00f99203e9f85cd305e4ee008250b

I downloaded around 1200 random audio files from freesound.org. Having no idea of their content, I made 16 different sound packs with them using this script. I played Serious Sam HD: TSE, switching between those packs. Here is a compilation video of my playthroughs: https://youtu.be/xz176X4MDz4

Exercise 2 - Be a wanderer in an open-world videogame:

> Choose a videogame of your liking, with an open world.

> Find a finished save file for the game (the whole world should be open to roam around freely, fast travel to any spot).

> If possible, mute anything in the game until you only hear the sounds of characters and objects / soundscapes of your surroundings. Our purpose is to prepare for a sterile aural experience without distraction.

The unwanted sound types usually are:

- music, soundtracks
- ui sounds: menu, notification, alert, anything distractive
- AI, NPC: people talking to each other in a city could be acceptable, but if there are entities/monster attacking randomly, it could sabotage the experience.
- footstep sounds: if our character is making too much noise by just walking (or the other NPCs following you), it might be better to turn it off.

Tips for removing sounds:

- check the game menu if different types of sounds can be volumed down,
- search for mods by community, which may help with muting spesific classes of sounds.
- find the audio file of that sound in the game data, and replace it with silence.

[6] https://store.steampowered.com/app/41014/Serious_Sam_HD_The_Second_Encounter/

- Search for cheat codes / trainers available for your game of choice, to check if there is any useful cheat which could make things easier for you (such as god mode, disabling ai interaction).

> Be a wanderer in wherever you are in the game. Avoid interacting with NPCs, walk between landscapes, cities, buildings; meditate, and explore the sounds around you. Think about the mechanisms of nature and societies in the game, based on what you are hearing (and seeing).

EXAMPLE: Skyrim Recordings

Before sharing my recordings with The Elder Scrolls V: Skyrim[7], I would like to mention some works, that gave me the idea to write and do this exercise.

Appropriated field recordings from temporary data sources - Philip Sulidae:

https://soundcloud.com/gruenrekorder/appropriated-field-recordings-from-temporary-data-sources-philip-sulidae

https://www.gruenrekorder.de/?page_id=15809

HOW BIG IS THE MAP in Watch Dogs 2? Walk Across the Map:

https://www.youtube.com/watch?v=cy_74IDaRQs

For The Elder Scrolls V: Skyrim; I found a finished save file, muted distractive / out of context sounds from sound settings, and used cheats to disable AI and give myself immortality; and I started wandering around and made 3 different recordings.

I would suggest the reader to listen to the audio recordings first, focusing on repetitions, movements and patterns; then watch the video versions to compare if they are seeing what they depicted in their minds.

Audio recordings: https://soundcloud.com/princesscamel/sets/being-a-wanderer-in-the-elder

Video recordings: https://www.youtube.com/playlist?list=PLL-JW4H9We1wVvoM8XDKey0fE6YRFdK0d

[7] https://elderscrolls.bethesda.net/en/skyrim

Exercise 3 – Compositions:

Choose a game. Break down the gameplay mechanics, imagine a performance in a videogame, find your solution for how to notate the user inputs, and write short etudes/compositions with limited material.

Examples:

1 – Etudes for Fighting Games

No 1: Projectiles

Duration: 1 round

Both players choose the same character that has the ability to throw fireballs.

When the round starts, both players start shooting fireballs to each other as syncronous as possible on the given speed pattern below.

If the game doesn't have 4 different versions of fireballs, players should optimize the pattern according to the inputs the game provides. For example: If the videogame doesn't have H, players should use M instead.

M M L H H H H H H H H EX L L L L H M H M L L L M H L M H EX M L H L M M M M H M H M L L L L L L L L L H H L H M M EX L H L L M M M M M H H H H H H L L L L M M H H L L M M L L H H M H M H M H EX

No 2: Footsie Grabs

Duration: 1 round

Players move or dash back and forth randomly (without crouching or jumping). When the characters get next to each other (in a range where they can grab each other), both players press grab at the same time.

After each 2-4 grabs, players can jump forward or backwards once.

No 3: Let's Chill Together

Duration: 1 round

In the beginning of the round, both players move their characters to a point of their choice, and crouch there for the rest of the round.

No 4: Fistfight

Duration: 1 match

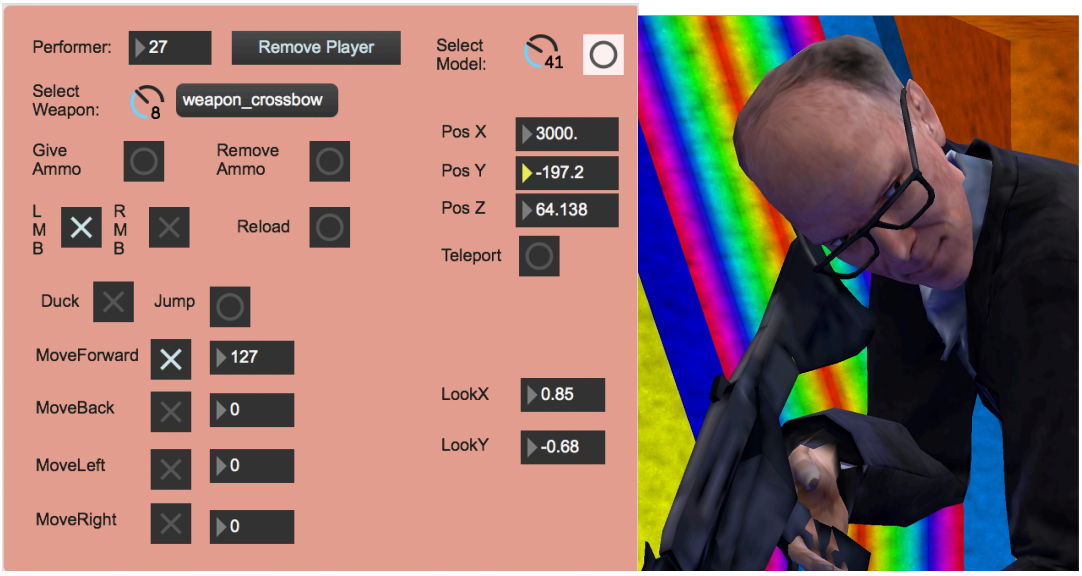
Players fight with only using punches (no special moves).

No 5: Imitation Game

Duration: 1 round

One of the players moves back or forth, crouches or jumps, or just stands; the other one imitates what the other one's movements as he/she sees.

PPPB (Pseudo Player Performer Bots)



[8]

A year before I started writing this text, I was curious about how turning a videogame into an interactive performative space would work as a medium to compose pieces for people to perform in.

In this chapter, I will talk about how this idea evolved into making PPPB (Pseudo Player Performer Bot(s)); a toolset I developed for "Garry's Mod" to make audiovisual compositions and performances. PPPB allows the user to send MIDI messages to Garry's Mod to create (up to 64 bots), and control them individually via a MIDI controller, or an external software.

I chose "Half-Life" to be my first game for experimenting, which is an iconic First Person Shooter game, that revolutionized the FPS genre at that time, and the videogames in general. Half-Life has a Multiplayer mode, where players can join a server and start shooting each other.

FPS games (especially classic ones) are mostly fast paced. While playing it, the player should be constantly aware of their surroundings, and react / aim fast to be the first to shoot others, before getting shot.

In these types of games, the player gets very fast visual / sonical response of their user inputs; therefore the controls can be approached as an instrument interface to quickly experiment with musical ideas.

I spent some time practicing Half-Life in "crossfire[9]", one of the most popular HL multiplayer maps. I studied player controls, map design, spawn points, item locations etc; hoping to get some answers to the questions I asked below to choose a notation system and structure compositions:

---

[8] http://berkozdemir.com/pppb

[9] https://combineoverwiki.net/wiki/Crossfire_(multiplayer_map)

Instrument

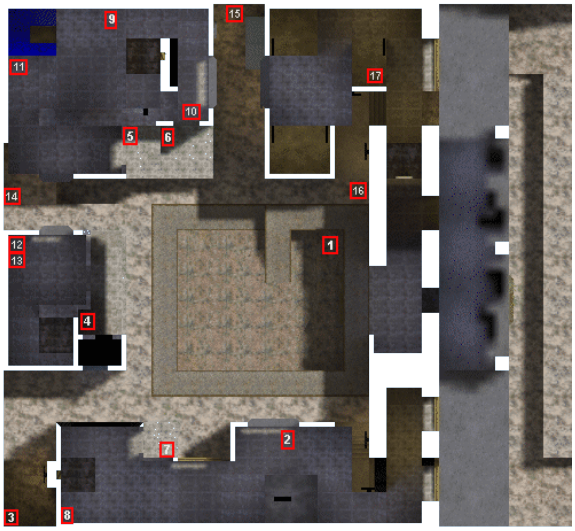
What are the controls/moveset given to the player by the game?

Move (forward, backwards, left, right), crouch/stand, walk/run, jump, look, change weapon, fire weapon, secondary use of weapon, reload weapon, interact with objects.

What kind of sounds do those moves generate? Are the sounds fixed with each move, or do they alternate with each trigger? What kind of musical gestures can be found with the combination of them?

Moves usually generate sounds played from an audio file associated with it. Amplitude, timbre and pitch is fixed, which leaves rhythm as the main element to experiment with musical ideas.

Space / Stage[10]



- How do the physics and the space of the game react to actions and movement of a player?

- How can the characters be distributed / moved in the space for spatialization?

- In which kind of environment will players perform? In their homes? A convention? A concert stage? A net-cafe?

- How will the audience see the in-game performance? Is it going to be from a performer's character's eyes, or a spectator's? Is it going to be projected on a wall, or streamed on internet?

Notation

- What kind of notation can I use to make the cleanest communication possible with performers, and express my musical ideas? How can I notate the moveset / gestures?

- Traditional notation - text notation - graphic notation

- Strict notation vs. open score?

- From where can the performers follow the score?

- From a paper, or the multiplayer server itself, which posts text intructions to each performer

---

[10] https://steamcommunity.com/sharedfiles/filedetails/?id=121904692

After going through brainstorming, I decided to switch the videogame from Half Life to Garry's Mod[11], which is a physics sandbox game based on Half Life 2's game engine, that provides tools for scripting, game mod creation, level/entity/character design and so on. Gameplay mechanics-wise, Garry's Mod is almost identical to Half-Life, and it has a better physics engine and tons of user created content online, which can be downloaded and used for creating new stuff.

My reason for switching to Garry's Mod was to use scripting tools and libraries in-game to build a server that is online 24/7; which:

> starts a new performance every "n" minutes

> players that join can choose to be a "performer" or "audience"

> is the stage and the notation medium for performers

> will send different instructions to each performer to do, within set time intervals

As I started working with this idea, I noticed that working with bots instead of humans would be a much easier way to experiment with player controls.

A bot is a NPC, which is used as replacement of human players in multiplayer games. They are mostly programmed by an algorithm to imitate a human player, and depending on how complex their AI is, they might be beaten easily, or give the player a hard time.

In my childhood, when I played Half-Life or Counter Strike at home, I would usually play it with bots, because I had limited access to internet at that time; which made it very hard for me to join the online game servers and play with real players. So I used to download different bots made by different people, and play the game with them. Some of the bots would have settings to adjust their behaviour, so I could change them for new experiences.

If I could create characters in game and control them like a real player in real time, I wouldn't have to think about distortion in communication between me "as a composer" and the performers, and I could easily test, prototype and record my ideas.

For instance; if I asked 8 players to shoot the gun with a fixed unison rhythm, then start phasing their shooting speed one by one slowly, I would need 8 people who are capable of doing it, and I would also have to consider the latency between server and each player. But I could try this idea easily by sending commands to each bot, and they would execute it perfectly.

I started learning the programming language to script Garry's Mod, Lua, went through all "bot" examples shared by community. I also wanted to provide a communication with these bots from an external source, using a protocol like MIDI or OSC. But until I could find a way to do it, I had to do my experiments within limited controls.

---

[11] https://gmod.facepunch.com/

Hello bots – https://www.youtube.com/watch?v=Zc2SOyyX6qA

Keypad mapping test – https://www.youtube.com/watch?v=qWVMd6Cv6YA

3 White Room Etudes:

I recorded these etudes before implementing MIDI, therefore I could only send monophonic commands to all bots via keyboard strokes. A command I send like shoot, jump, would be sent to all bots simultaneously (sometimes with a little big of latency, especially if the summoned bot amount is too many, like 60-100). So I couldn't control them individually.

These etudes worked with some simple rules;

Bots are spawned in the white room of map "gm_construct", with predetermined coordinates, angles and weapons (forming squares). The moment they are born, they start shooting. Spawning them in different time internals generates different polyrhythms.

I have the control on changing the weapons, and starting / stopping shooting. I played with these controls to make 3 different recordings.

The etudes can be listened / watched from these links:

Music album: https://berkozdemir.bandcamp.com/album/3-white-room-etudes-for-pppb

White Room Etude #01: https://www.youtube.com/watch?v=eUFSybUsSno

White Room Etude #02: https://www.youtube.com/watch?v=w2GLPufYN_A

White Room Etude #03: https://www.youtube.com/watch?v=gHhZwwMSSTk

Apparently, there was a MIDI module already existing for Garry's Mod called gmcl_midi[12], which allows users to send MIDI messages to the game. As I saw, the only use case of this module was playing piano or other instruments. Since my intention to use MIDI was for controlling every individual bot dynamically, I had to structure a new mapping system to assign each possible property of bots I could control with MIDI notes and CCs.

MIDI Keyboard Test Video - https://www.youtube.com/watch?v=7w2HnKfjsTA

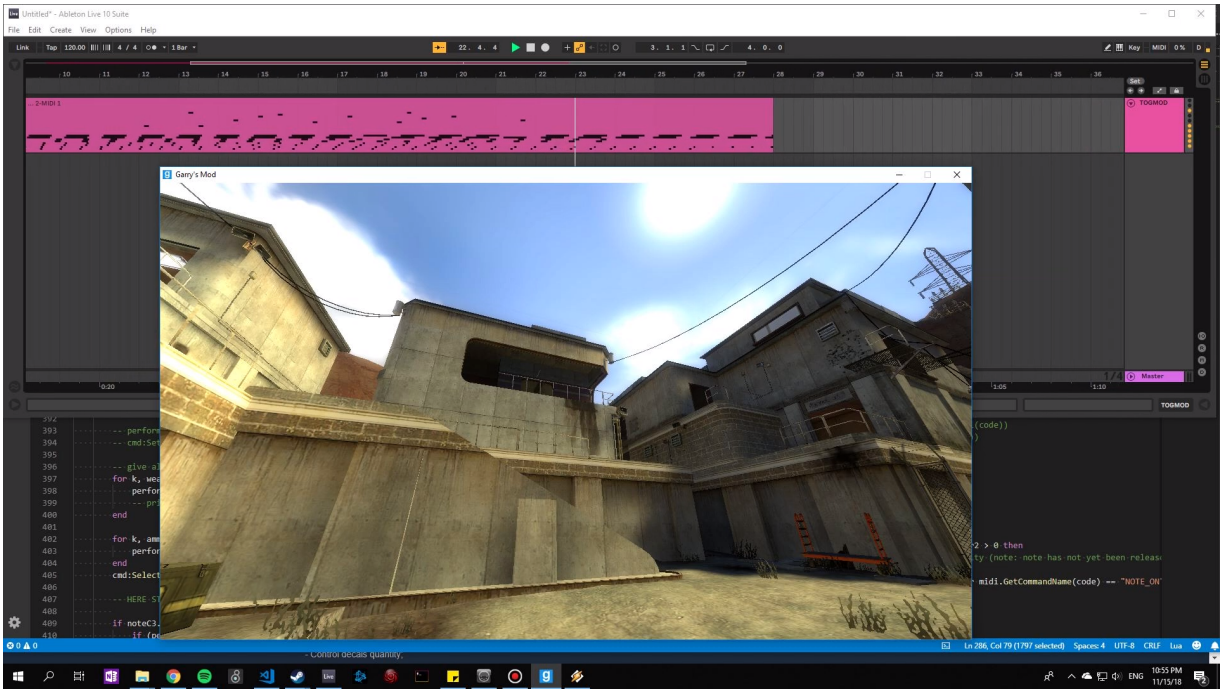
After some work; I built the MIDI mapping inside Garry's Mod, and a controller interface in Max (https://cycling74.com) to make things easier for me. Now I had the bots ready to do anything I want them to do.

With PPPB, I worked with algorithms, chance, probability, phasing, rhythmic patterns, notation, improvisation, communication and syncronisation of multiple media.

[12] https://github.com/FPtje/gmcl_midi

Here are the video links to the pieces / compositions / experiments I made:

Awesome Crossfire Solos – https://www.youtube.com/watch?v=1He1nG98Rec



Clapping - https://www.youtube.com/watch?v=wmDZmhrUUfM

Finger Snapping #1 – https://www.youtube.com/watch?v=dPegR2iyUgE

Finger Snapping #2 – https://www.youtube.com/watch?v=u1zjTt4h38U

Quartet 1 – https://www.youtube.com/watch?v=h82d2kINH8M

Accelerando - https://www.youtube.com/watch?v=B7sHTj7wwDc

Random Roulette - https://www.youtube.com/watch?v=xn1zf_N36Zg

Algorithmic Study - https://www.youtube.com/watch?v=7p2aUs2JblQ

The source code for PPPB, and my log book of development process can be found on Github: https://github.com/vortextemporum/pppb

OUTRO

This whole process of research and work has proven me that videogames can offer much more than just gaming experience, and work well as a composional and performative material.

Although this text covers only some spesific elements of a few videogames I worked with; this approach can be extended further with any other videogame genres, and artistic disciplines.

Or even further; not only with videogames, but anything we use on our computers.

Anything we use or consume on our computers, either is a software, or runs with a software.

We can deconstruct their elements like we did to videogames, and transform/ repurpose them for our creativity.

We use interfaces to interact with software. This interface is an instrument. We can play this instrument, write a script or algorithm that plays it for us, or connect another software's output to map controls.[13]

Websites, softwares, services, digital media, technologies... There are lots to discover and play.

[13] Codebending: http://www.paperkettle.com/codebending/
Illucia: http://www.illucia.com/

BONUS

There was a project I wanted to finish so much last year, but while I was working on it, I got stuck at some point, then I stopped working on it.

>>>>>>>>>> ESCAPE FROM LORELEI <<<<<<<<<<

https://www.urbandictionary.com/define.php?term=Softlock

I wanted to build a machine that continuously tries to escape Lorelei's softlock in Pokemon Blue, as shown in this video:

"How to Escape Lorelei's Game Ending Softlock". https://www.youtube.com/watch?v=CCIsivwN8aw

An automation script running with a GameBoy emulator does this loop:

> Use move "RAGE"
> If move hits, reset
> If move misses, continue waiting
> Count every try
> Store biggest row of misses

And this loop could be run in a small computer (maybe a Raspberry Pi) 7/24 until the move misses 20 times in a row.

It is a pathetic attempt, because the chances of missing the move 20 times is (1 / 256) ^ 20, and probably no one in their lifespan will see it happen.

Hopefully I will find answers to technical difficulties, then I can continue working on.

ALL LINKS

PAGE 9:

> Wikipedia Page | Speedrun -> https://web.archive.org/web/20200411141532/https://en.wikipedia.org/wiki/Speedrun

> Half-Life 2 Developers React to 50 Minute Speedrun -> https://www.youtube.com/watch?v=sK_PdwL5Y8g

PAGE 10:

> [LONGPLAY] NES - Super Mario Bros (HD, 60FPS) -> https://www.youtube.com/watch?v=1qcTwuKozs8

> Wikipedia | Musical Form (Labeling) -> https://web.archive.org/web/20200411141518/https://en.wikipedia.org/wiki/Musical_form#Labeling_procedures

PAGE 12:

> Youtube Search | "replaced all sounds in with" -> https://www.youtube.com/results?search_query=replaced+all+sounds+in+with

PAGE 14:

> Wikipedia Page | Serious Sam -> https://web.archive.org/web/20200411142947/https://en.wikipedia.org/wiki/Serious_Sam

PAGE 15:

> Steam Page | Serious Sam HD: TSE -> https://web.archive.org/save/https://store.steampowered.com/app/41014/Serious_Sam_HD_The_Second_Encounter/

> SERIOUS SOUND RANDOMIZER GUIDE -> https://gist.github.com/vortextemporum/18a00f99203e9f85cd305e4ee008250b

> I REPLACED ALL SOUNDS IN SERIOUS SAM HD: THE SECOND ENCOUNTER WITH RANDOM SOUNDS - https://www.youtube.com/watch?v=xz176X4MDz4

PAGE 16:

> Skyrim Official Page -> https://elderscrolls.bethesda.net/en/skyrim

> Appropriated field recordings from temporary data sources | Philip Sulidae ->

https://soundcloud.com/gruenrekorder/appropriated-field-recordings-from-temporary-data-sources-philip-sulidae

https://web.archive.org/web/20200411135808/https://www.gruenrekorder.de/?page_id=15809

> Youtube Video | HOW BIG IS THE MAP in Watch Dogs 2? Walk Across the Map -> https://www.youtube.com/watch?v=cy_74IDaRQs

> SKYRIM AUDIO RECORDINGS -> https://soundcloud.com/princesscamel/sets/being-a-wanderer-in-the-elder

> SKYRIM VIDEO RECORDINGS -> https://www.youtube.com/playlist?list=PLL-JW4H9We1wUvoM8XDKey0fE6YRFdK0d

PAGE 19:

> PPPB -> http://berkozdemir.com/pppb

> Crossfire map -> https://web.archive.org/web/20200411140258/https://combineoverwiki.net/wiki/Crossfire_(multiplayer_map)

PAGE 20:

> Half Life Spawn System -> https://web.archive.org/web/20200411135949/https://steamcommunity.com/sharedfiles/filedetails/?id=121904692

PAGE 21:

> Garry's Mod -> https://gmod.facepunch.com/

PAGE 22:

> PPPB - Hello bots -> https://www.youtube.com/watch?v=Zc2SOyyX6qA

> PPPB - Keypad mapping test -> https://www.youtube.com/watch?v=qWUMd6Cv6YA

PAGE 23:

> 3 White Room Etudes for PPPB | Berk Özdemir -> https://web.archive.org/web/20200411135052/https://berkozdemir.bandcamp.com/album/3-white-room-etudes-for-pppb

> PPPB - White Room Etude #01 -> https://www.youtube.com/watch?v=eUFSybUsSno

> PPPB - White Room Etude #02 -> https://www.youtube.com/watch?v=w2GLPufYN_A

> PPPB - White Room Etude #03 -> https://www.youtube.com/watch?v=gHhZwwMSSTk

Page 24:

> gmcl_midi module for Mod -> https://github.com/FPtje/gmcl_midi

> PPPB - MIDI Keyboard Test -> https://www.youtube.com/watch?v=7w2HnKfjsTA

> Max Official Page | https://web.archive.org/web/20200411134946/https://cycling74.com/products/max

Page 25:

> PPPB - Awesome Crossfire Solos —> https://www.youtube.com/watch?v=1He1nG98Rec

> PPPB - Clapping -> https://www.youtube.com/watch?v=wmDZmhrVUfM

Page 26:

> PPPB - Finger Snapping #1 -> https://www.youtube.com/watch?v=dPegR2iyUgE

> PPPB - Finger Snapping #2 -> https://www.youtube.com/watch?v=u1zjTt4h38U

> PPPB - Quartet 1 -> https://www.youtube.com/watch?v=h82d2kINH8M

Page 27:

> PPPB - Accelerando -> https://www.youtube.com/watch?v=B7sHTj7wwDc

> PPPB - Random Roulette -> https://www.youtube.com/watch?v=xn1zf_N36Zg

> PPPB - Algorithmic Study -> https://www.youtube.com/watch?v=7p2aUs2JbIQ

> PPPB Source Code -> https://github.com/vortextemporum/pppb

Page 28:

> Codebending FAQ -> https://web.archive.org/web/20200411160618/http://www.paperkettle.com/codebending/

> Illucia | Chris Novello -> http://www.illucia.com/

Page 29:

> Softlock -> https://web.archive.org/web/20200411144532/https://www.urbandictionary.com/define.php?term=Softlock