Click and Run

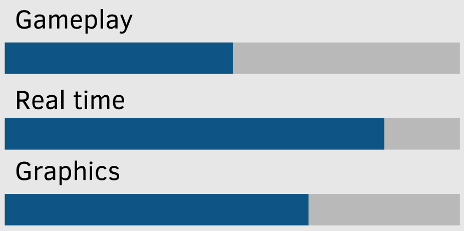
A canabalt-like game

3 people

Project Description

In this game, a player-controlled character is constantly running through a world, always in the same direction. The only control that the player has is to make the character jump to avoid various sorts of threats. The objective is to run for as long as possible. The world is generated procedurally and on the fly.

Skills



(*)[The skill scale is from 0 (Fundamental Awareness) to 6 (Expert).]

Level 1 You may now pursue to the level 1 of the project.

⋆⋆ Procedural world generation
Generate an infinite world with a fair amount of variety. The world must consist (at least) of floors and steps and holes.

⋆⋆⋆ World display and start screen
The game should be able to display (part of) the world. The start screen should be scrolling from left to right through a procedurally generated world.

⋆ Player
A character is running through an empty world, and it can jump and fall; it must not be able to go through walls and floors.

⋆ Complete game mechanics
Put the previous two items together and detect death conditions, at least collision but perhaps also falling out of the screen.

⋆ Animation
Animate the character, its jumps, and perhaps its death(s).

⋆ Background
Generate an infinite background for the world, and display it with a slower scroll, simulating a parallax effect.

⋆ Score
Compute a score during the game, increasing (only) when an obstacle is avoided.

⋆ Score UI
Display the score during the game, and have a leader board which prompts for a player name when a new high-score is reached.

Click and Run

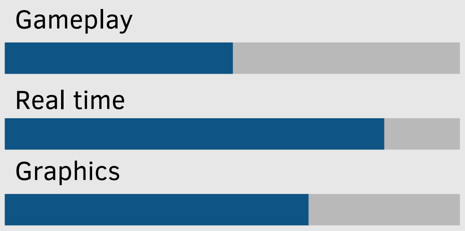
A canabalt-like game

3 people

Project Description

In this game, a player-controlled character is constantly running through a world, always in the same direction. The only control that the player has is to make the character jump to avoid various sorts of threats. The objective is to run for as long as possible. The world is generated procedurally and on the fly.

Skills



(*)[The skill scale is from 0 (Fundamental Awareness) to 6 (Expert).]

Level 2 Level 1 must be unlocked to read this section

⋆ **Developper Documentation** Required for lvl 2 validation
Document your project (not necessarily only in the source code) so that a newcoming developper could understand and contribute to the code.

⋆ **Release** Required for lvl 2 validation
Produce a release as a source archive or git tag. The release files should have up-to-date README and INSTALL files and more generally allow anyone to deploy the application.

⋆ **Adaptative difficulty**
Introduce new difficulties, in the form of new items or landscape features, when the score reaches predefined levels.

⋆ **Integration test**
Test the run of a predefined game (some basic IA with a predefined map)

⋆ **Test procedural generation**
Generate images of complete level generations, that can be bulked analyzed by a human to juge the effectiveness of the generation. Test that there is always a winning strategy.

⋆ **Adaptative speed**
Change the scrolling and running speed depending on the score or its derivative.

⋆⋆ **Items**
Add special items that one that instantly kills the player, one that gives an extra life (displayed on a life counter on screen), on that kills the player unless an antidote is picked within 20 seconds. Add items for temporarily slowing down or speeding up...can be picked (by running over them): that affects the player life, the gravity, the game speed, the visual rendering of the screen

⋆ **Visual effect**
Add temporarily visual effects caused by some event.

⋆ ⋆ ⋆ **Enemies**
Add enemies to the map. A sort of enemy should walk towards the player character, another jump following a predictable pattern.

⋆⋆ **Two player mode**
Add a two-player mode where players are on the same machine, each one controlling a character.

⋆ ⋆ ⋆ **Replay**
Add the possibility to replay past games. For instance, high score runs could be displayed on the start screen.

⋆⋆ **Performances**
Evaluate the performances of the game, both in space and time, using profiling, and demonstrate the limits at high speed with many enemies.

Minisoap

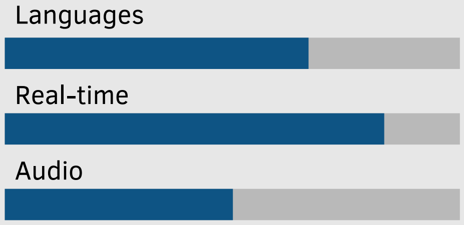
A modular audio processor

3/4 people

Project Description

An audio stream processor that can create and transform audio streams, featuring basic synthesis capabilities as well as file and soundcard input/output. It is designed modularly, allowing the user to specify its own processing pipeline.

Skills



(*)[The skill scale is from 0 (Fundamental Awareness) to 6 (Expert).]

Level 1 You may now pursue to the level 1 of the project.

⋆⋆⋆	Stream processors Design a notion of (audio) stream generator that works with buffers (not samples) for efficiency, and has a notion of track. A stream may become unavailable at the end of a track. It may become available again after some delay.
⋆	Basic generators Implement a silence generator, and a sine generator parameterized by its frequency and amplitude.
⋆	Scheduling operators Implement fallback and rotation operators. Both take a list of input streams. The first one repeatedly plays a (complete) track from the first available source. The second one plays a track from an input, and then the next one from the next available input (rotating back to the beginning of the list if needed.)
⋆	Mixing Implement an operator that mixes its input.
⋆⋆	Input/Output Implement a source that reads from the soundcard, and make it possible to output a stream processor on the soundcard.
⋆⋆	Files Implement input/output from/to one common audio file format.
⋆⋆	Playlists Implement a stream generator which, given a playlist file, plays its files sequentially. It should offer shuffling and repeat options.
⋆⋆	Transitions Implement transitions between tracks: the stream should be modified at the end of a track and at the beginning of the next one. At least fading should be supported.
⋆⋆⋆	Language Make it possible for users to describe an audio processing pipeline in a simple domain-specific language.

Minisoap

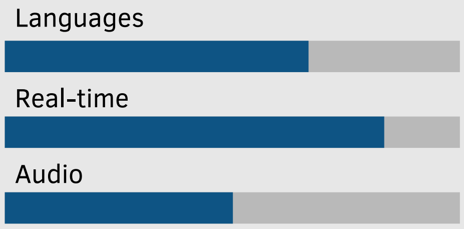
A modular audio processor

3/4 people

Project Description

An audio stream processor that can create and transform audio streams, featuring basic synthesis capabilities as well as file and soundcard input/output. It is designed modularly, allowing the user to specify its own processing pipeline.

Skills



(*)[The skill scale is from 0 (Fundamental Awareness) to 6 (Expert).]

Level 2

Level 1 must be unlocked to read this section

⋆ **Developper Documentation** — Required for lvl 2 validation
Document your project (not necessarily only in the source code) so that a newcoming developper could understand and contribute to the code.

⋆ **Release** — Required for lvl 2 validation
Produce a release as a source archive or git tag. The release files should have up-to-date README and INSTALL files and more generally allow anyone to deploy the application.

⋆⋆ **Sharing**
The stream processing engine should properly handle situations where a unique stream is being consumed as input of several other streams. In such a situation, identical samples should be sent to each consumer at each instant, copying/caching data if necessary.

⋆ **Vumeter**
A vumeter feature should make it possible to visualize in realtime the intensity of the stream at any given point in the stream processing graph.

⋆⋆⋆ **Self-documenting UI**
The user interface should have a help feature providing a list of available operators with their documentation. It should be designed so that adding a new operator to the system (including its documentation) involves the modification of only one file.

⋆⋆ **Realtime parameter changes**
The parameters of some operators (e.g. the amplification factor of an amplification operator) should be updatable in realtime by the user. This should be possible at least in the console (text mode) or through a graphical UI (e.g. using libio).

⋆⋆ **Realtime pipeline updates**
The user should be able to modify the streaming production pipeline, by typing in commands, while the streams are running.

Magic

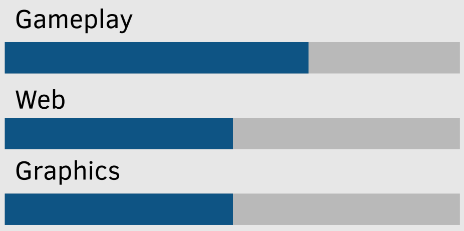
A collectible card game

4 people

Project Description

In this game, two players fight each others, drawing cards from a deck. Cards can be monsters, spells, enchantments,...

Skills



(*)[The skill scale is from 0 (Fundamental Awareness) to 6 (Expert).]

Level 1 You may now pursue to the level 1 of the project.

⋆⋆	Basic game mechanics There exist a notion of players, monster cards, and decks. The logic for a simple game is developped.
⋆⋆	Game server A server allows to interact with the core.
⋆⋆	Duel arena display A web interface allows one player to interact with the server and play a game, if another player is connected.
⋆⋆⋆	Special cards There exist multiple kind of cards: instance objects, effects, counters. There should exist a reasonble variety of available cards, with non trivial interactions between them. Interactions between cards should be well defined, and never result in a conflict.
⋆	The big league Players must register an account on a server. If the server crashes, data is preserved. They can manage their decks, obtaining new cards when winning games. A global ranking is displayed.
⋆	Animations The arena displays some animations for some actions.
⋆	AI Two AIs must be implemented. One perfectly random, and one which loses to the random one with negligible probability.

Magic

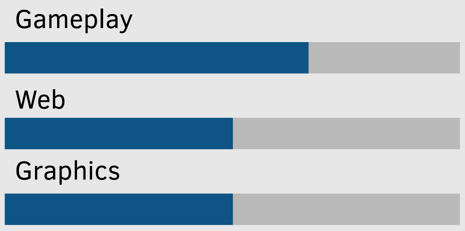
A collectible card game

4 people

Project Description

In this game, two players fight each others, drawing cards from a deck. Cards can be monsters, spells, enchantments,...

Skills



(*)[The skill scale is from 0 (Fundamental Awareness) to 6 (Expert).]

Level 2

Level 1 must be unlocked to read this section

⋆ **Developper Documentation** — Required for lvl 2 validation

Document your project (not necessarily only in the source code) so that a newcoming developper could understand and contribute to the code.

⋆ **Release** — Required for lvl 2 validation

Produce a release as a source archive or git tag. The release files should have up-to-date README and INSTALL files and more generally allow anyone to deploy the application.

⋆⋆ **Integration testing of AI**

Same as above, but with integration test that verifies that the not-random player is better than the random one, by playing a large number of runs.

⋆⋆⋆ **Variant game**

It should be possible to adapt the system for some variant of the game (new set of cards, including some new features) that will be specified after the initial game has been presented.

⋆⋆ **Client performance**

The client reactivity (animations, etc.) must be satisfying. Its memory consumption should be evaluated using a javascript profiler, whose output will be presented and justified.

⋆⋆ **Failure resilience**

The game should be resilient to network lags, in particular avoiding desynchronizations. No tests are required, but the design will have to be presented and justified.

Rythm and Run

A rythmic canabalt-like game

5 people

Project Description

In this game, a player-controlled character is constantly running through a world, always in the same direction. The only control that the player has is to make the character jump to avoid various sorts of threats. The objective is to run for as long as possible. The world is generated procedurally and on the fly.

Skills

Gameplay	▇▇▇▇▇▇▇▇▇▇▇▇░░░░░░░░
Real time	▇▇▇▇▇▇▇▇▇▇▇▇▇▇▇▇▇░░░
Graphics	▇▇▇▇▇▇▇▇▇▇▇▇▇▇░░░░░░

(*)[The skill scale is from 0 (Fundamental Awareness) to 6 (Expert).]

Level 1 You may now pursue to the level 1 of the project.

⋆⋆ Procedural world generation
Generate an infinite world with a fair amount of variety. The world must consist (at least) of floors and steps and holes.

⋆⋆⋆ World display and start screen
The game should be able to display (part of) the world. The start screen should be scrolling from left to right through a procedurally generated world.

⋆ Player
A character is running through an empty world, and it can jump and fall; it must not be able to go through walls and floors.

⋆ Complete game mechanics
Put the previous two items together and detect death conditions, at least collision but perhaps also falling out of the screen.

⋆ Animation
Animate the character, its jumps, and perhaps its death(s).

⋆ Background
Generate an infinite background for the world, and display it with a slower scroll, simulating a parallax effect.

⋆ Score
Compute a score during the game, increasing (only) when an obstacle is avoided.

⋆ Score UI
Display the score during the game, and have a leader board which prompts for a player name when a new high-score is reached.

⋆⋆⋆ Rythmic analysis
Procedural level generation should be impacted by some musical analysis of the background soundtrack, in some way or another.

⋆⋆⋆ Story mode
The game should feature a mode where a story plays out: this means that some events should trigger e.g. the display of dialogues, the move to another level, the addition of a new goal (catch an item, reach some number of points, kill a monster...) etc.

Rythm and Run

A rythmic canabalt-like game

5 people

Project Description

In this game, a player-controlled character is constantly running through a world, always in the same direction. The only control that the player has is to make the character jump to avoid various sorts of threats. The objective is to run for as long as possible. The world is generated procedurally and on the fly.

Skills

Gameplay	3
Real time	5
Graphics	4

(*)[The skill scale is from 0 (Fundamental Awareness) to 6 (Expert).]

Level 2

Level 1 must be unlocked to read this section

⋆ Developper Documentation — Required for lvl 2 validation
Document your project (not necessarily only in the source code) so that a newcoming developper could understand and contribute to the code.

⋆ Release — Required for lvl 2 validation
Produce a release as a source archive or git tag. The release files should have up-to-date README and INSTALL files and more generally allow anyone to deploy the application.

⋆ Integration test
Test the run of a predefined game (some basic IA with a predefined map)

⋆ Test procedural generation
Generate images of complete level generations, that can be bulked analyzed by a human to juge the effectiveness of the generation. Test that there is always a winning strategy.

⋆⋆ Items
Add special items: one that instantly kills the player, one that gives an extra life (displayed on a life counter on screen), on that kills the player unless an antidote is picked within 20 seconds.

⋆⋆⋆ Enemies
Add enemies to the map. A sort of enemy should walk towards the player character, another jump following a predictable pattern.

⋆⋆ Shield
An item should provide a shield to the player, in the form of some number of bullets rotating around the player. Collisions with enemies should be precisely computed, taking the bullets into account. A collision with a bullet kills only the bullet and not the player.

⋆⋆ Visual effect
Add temporary global visual effects caused by some event: an item should trigger a rotation of the screen, another should make the rendering black and white.

⋆⋆ Performances
Evaluate the performances of the game, both in space and time, using profiling, and demonstrate the limits at high speed with many enemies.