1 Week 1 (6/20/16 - 6/26/16)

1.1 Tasks

- Idea generation
- Early drafts
- Early game prototype

1.2 Idea generation

Required constraints:

- Game
- Heartbeat or breathing sensor
- Physical computing
- Q-learning
- Affective computing or soma-aesthetics

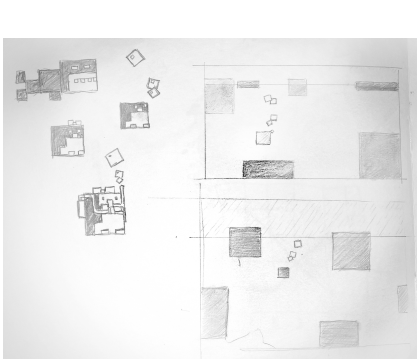
1.2.1 Final Concept: Flappy Penguin (working name)

After some brain storing we settled on a game in the spirit of *Flappy Bird* featuring:

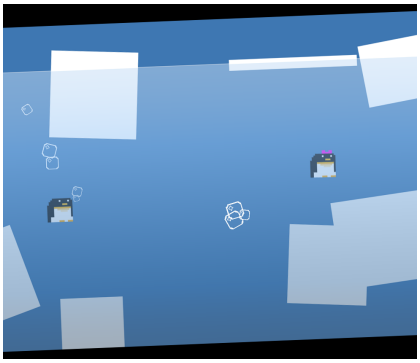
- "Protagonist" is a penguin
- Side-scrolling movement
- Ice blocks entering the stage as obstacles
- *Physical object* controls penguin movement
- Breath-meter indicates penguins remaining breath
- User breathing while the penguin is under water reduces the breath-meter
- User breathing while the penguin is at the water surface replenishes breath-meter
- Empty breath meter means death by suffocation, i.e. game over
- Additional air bubbles can be collected under water to increase breath
- *Q-learning* used for either placement of air bubbles of a second, computer-controlled penguin
- Visually using the style of *Thomas was Alone*

1.3 Early drafts

See Fig. 1



(a) Sketches

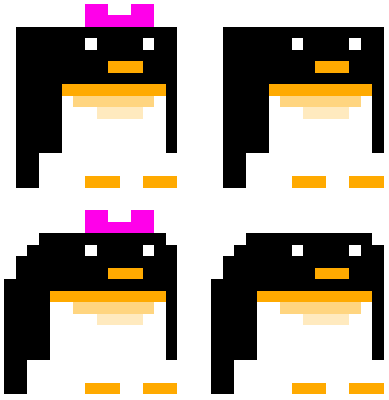


(b) Mockup

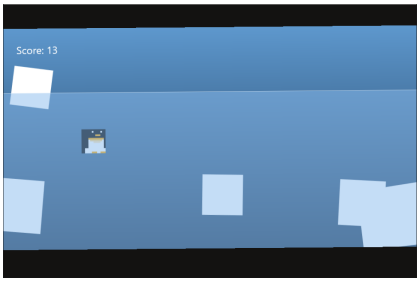
Figure 1: Concept

1.4 Early game prototype

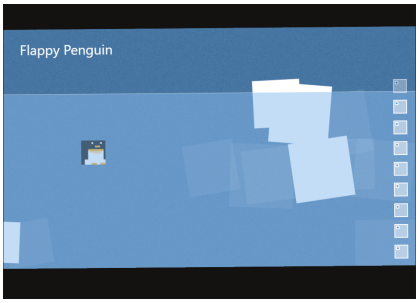
See Fig. 2



(a) Early game assets



(b) Prototype using first few assets



(c) Prototype using first few assets

Figure 2: Screenshots of different iterations

2 Week 2 (6/27/16 - 7/3/16)

This weeks tasks:

- Incremental improvements of the game
- Breathing sensor integration
- Bend sensor integration

2.1 Game improvements

By now the game includes the following components

(a) Final game

(b) Victory screen

- Basic scene layout
- Jumping by keyboard input
- Random obstacle generation
- Obstacle collision
- Breath and Breath GUI
- Pausing

Using a*NodeJS*server data is sent from the*Arduino*to the game via *Socket.IO*.

With this we are now able to decrease the breath-meter when the user breathes while the penguin is beneath the water surface and increase it when it is above.

2.2 Integration of sensors

With the game being designed from scratch for sensor integration, moving data from Arduino to the web-page was very straight forward.

We agreed on a simple JSON based protocol were messages had a type to distinguish between jumping and breathing input.

We also distinguish between breathing in an out though it has no application in the game.

3 Week 3 (7/4/16 - 7/10/16)

3.1 Tasks

- QLearning

3.2 QLearning

Q-learning is used for a second penguin swimming away from the player penguin. This penguin swims at a steady pace unless it bumps into an ice-block. This will slow it down, allowing the chasing player penguin to catch up. The controls of this second penguin are learned with the goal of escaping the player.

With a game such as Flappy Bird or Flappy Penguin, the state space is exceptionally large. In theory all obstacle positions, player position and in our version also opponent position and breath can factor into the full game state.

We decided that for the way we train our second penguin, properly dodging obstacles, player position and breath were irrelevant. Furthermore only obstacles in front of the computer player were important to it.

Thus we discretized the space in front of it into 16 sample points, for we encoded in binary whether there was an obstacle at this position or not.

```
var state = 0;
foreach (samplePoint) {
    if (obstacle at samplePoint) state = state | 1;
    state = state << 1
}
```

We also had to include the position of the learning penguin for which we used some of the remaining 16 Bits.

We discretized it to 64 values, 6 Bits, which we appended to the obstacle description.

```
state = state << 6;
state = state | discreetPenguinPosition;
```

Thus we were able to encode the state into a 22 Bits Integer. A state space of 2^22 state is of course still exceptionally large, but in reality most states will never occur.

4 Week 4 (11/7/15 - 18/7/16)

4.1 Tasks

- Fine tuning
- Presentation

4.2 Fine tuning

The last few improvements to the game were:

- Adding air bubbles as visual cue when breathing under water.
- A shattering animation for ice-blocks when colliding with them
- Music integration

- Win and Game Over screens

4.3 Presentation

The presentation section assigned to me was concerning q-learning.

Since all listeners should have a solid understanding of how q-learning works and the vast majority being familiar with implementation details, only few parts of this topic were available as sufficiently interesting presentation subjects.

The implementation details in JavaScript using the idiomatic way of callback functions and a second third runtime loop (aside from the rending and update loops) seemed not overly interesting, since they cover more JavaScript than q-learning.

Thus we decided to cover one of the main challenges of q-learning during the presentation, the state description, which for a game as Flappy Penguin is highly non trivial. However, as previously discussion, we managed to encode the state into 22 Bits, thus producing a state space of size $\ll 2^2 2$.

Unfortunately, shortly before the presentation our penguin-controller broke. We tried to quick-fix this by mapping the jump key on the presenter we used during the presentation. This way we still had a sort of physical controller and didn't need to resort to using the traditional way, the keyboard.

What we did not anticipate was that another button on the presenter was by default mapped to the$F5$key, which, unfortunately, refreshed the page every time accidentally pressed.

We did manage however to present and demo the functionality as well as the great trailer of game.