Projective Game Platform

HTI-Project 2015 – Final Presentation

Lucas Pires Camargo

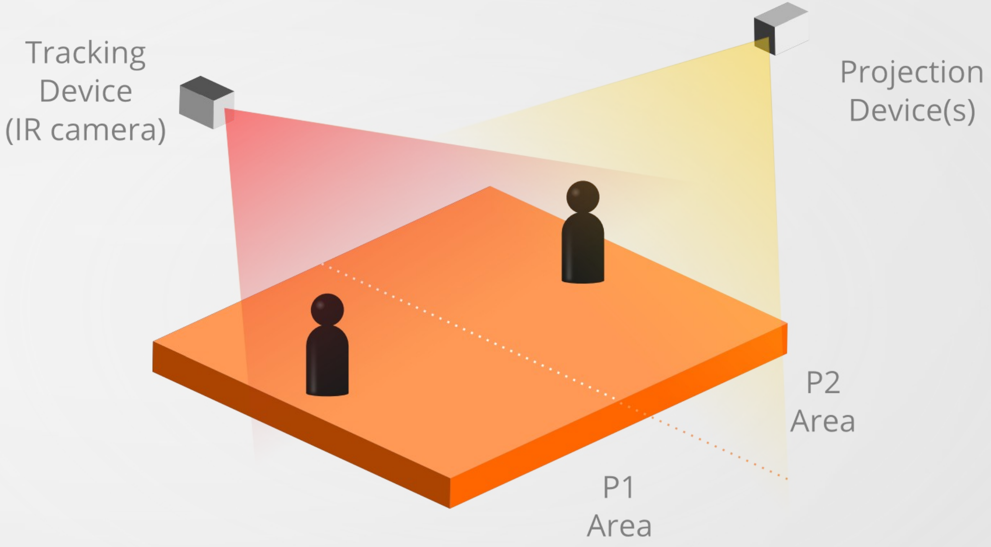
420145

TAUCHI
TAMPERE UNIT FOR
COMPUTER-HUMAN INTERACTION

TAMPEREEN
YLIOPISTO

Main Idea

- Public Space Gaming Platform, based on image projection and motion capture.
- Supports two players for competitive or cooperative play.



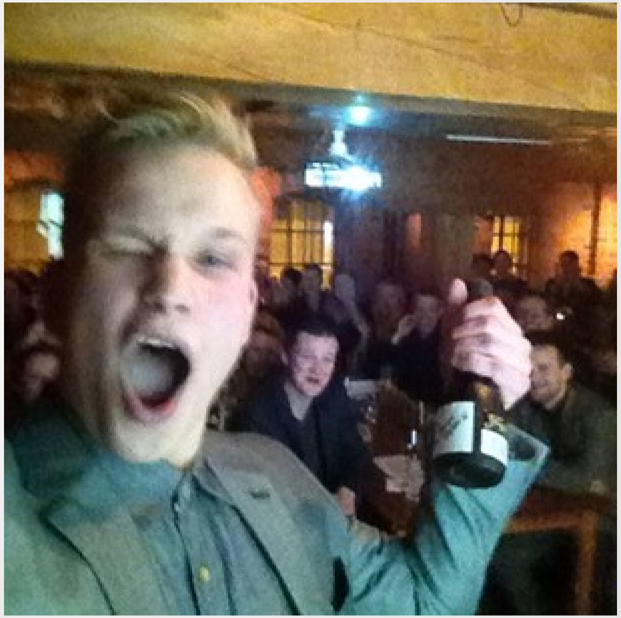
IR camera tracks players positions on field
Projection devices create image on playfield

Usage Environment

- City parks and malls, kid's playgrounds, etc
- Harsh sunlight might be a problem, shadowed or indoor areas are preferred.

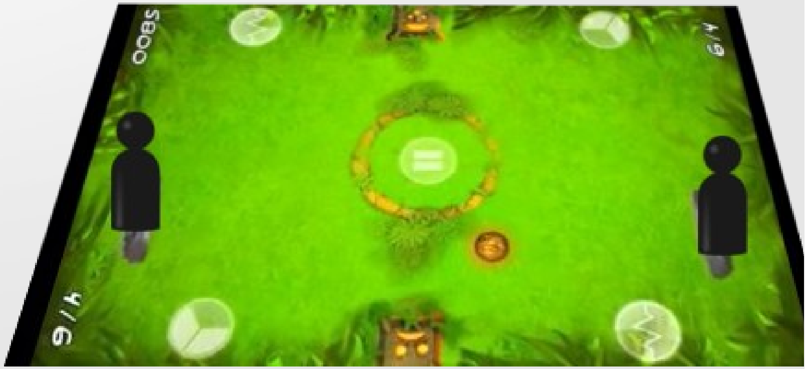
User persona: Juha

- Juha is a 18-year old nursery student.
- He likes spending time with friends, partying, and chilling out with his boyfriend.
- He is a very expansive person and is not afraid going out of his way to have fun.
- For that (and maybe other reasons) he does not consider himself very finnish.



Usage Scenario 1

- Jonne is with his friends and they are going for a beer.
- In the middle of the way, they find a public game.
- At first, they are shy to engage the device, but challenge each other for a match anyway.
- In pairs, they step on the surface for a match of Pong.
- Each player uses ther body position to control the paddles.
- First player to score five balls wins the game.

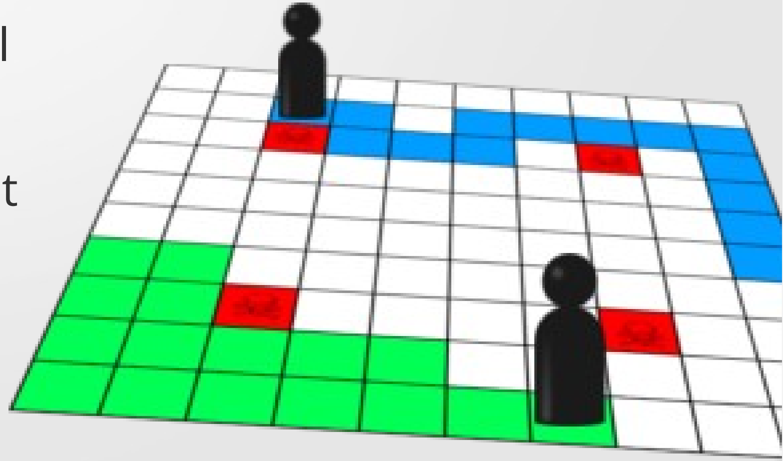


Users persona: Marja

- Marja is a young school-age girl living in Helsinki.
- She is walking around the mall with her parents and cousin of same age.
- They find a public game space and the kids jump straight into it.
- They get competitive and don't want to leave.
- The parents sigh :)

Usage Scenario 2

- Marja and her cousin step on the game board.
- They are presented with a grid of squares.
- They have to step on the squares to color them with "their" color. There are audio cues and nice graphics.
- The player that steps on more squares wins the round.
- As the rounds progress, some squares might be hurtful to the player's score.
- At the end of the game, the player who scored the most rounds wins. The game also ends when both leave the arena for a while.
- Both kids step out of the surface. Game resets.



Hardware: Input

- IR cameras for tracking of players in the field.
  - The ones in the lab are suficient for testing.
  - Kinect is a sufficient example.

Hardware: Output

- Standard projectors for image output.
  - High resolution and high brightness is desired.
  - The ones we have available should suffice for testing.
  - 4 would be optimal (3 is geometrically sound, but problems with distortion and unoptimal area usage)
- Audio amplifier

4 x

Hardware: Processing and Other Stuff

- Standard desktop computer.
  - Needs decent graphics card
  - Multiple video outputs desired (4)
- A bright, clear surface for projection on the floor
- Finally, support hardware for managing cables, protecting and hiding hardware, supplying power, etc

Software: Considerations

- We need two main things:
  - Track players in the field
  - Multiple output of the same scene with different transformations
- Other software components, like audio output and game engine, are pretty standard
- For this task, I choose QtQuick + OpenNI 2 + NiTE 2
- Running on common desktop stack (Linux, can be Windows)

Software: QtQuick

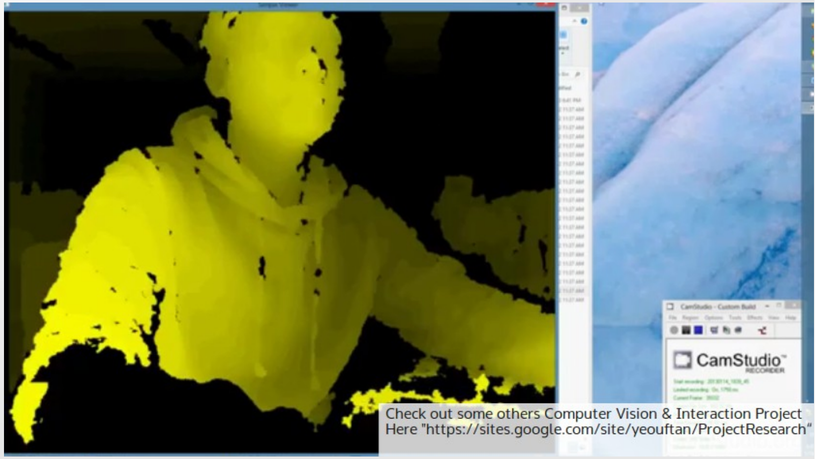
- A component of the ubiquitous Qt Frameworks
- Allows for accelerated 2D graphics with good performance
- Multiple outputs of same surface with transformations is trivial
- Many facilities for particles and graphical effects
- Rapid development (experience and resources)
- C++ is always available for integration with any other libraries.

Code less.
Create more.
Deploy everywhere.

Software: OpenNI 2 + NiTE 2

OpenNI® SDK 2

- Made specifically for use with user tracking in depth images.
- OpenNI has source available, several tutorials, examples.
- NiTE is now defunct, but still works well with OpenNI 2.
- Another implementation could use other libraries as backend.



Check out some others Computer Vision & Interaction Project Here "https://sites.google.com/site/yeouftan/ProjectResearch"

Basic interaction: Game Start and Stop

- The presence of the players is enough to trigger game start
- Depending of the game, it might require presence in specific places in the surface
- The game resets if everyone leaves

Basic Interaction: Cues

- The game must convey information in the most intuitive way possible.
  - Color coded game elements for easy understanding of roles
  - Instantly recognizable symbols (e.g., skulls for danger)
  - Common game tropes (square grids, ships, paddles, balls)
  - Avoid lots of text (viewing orientation is not usually fixed)
- Users are assumed to have never seen the game before.

Exceptional Interactions

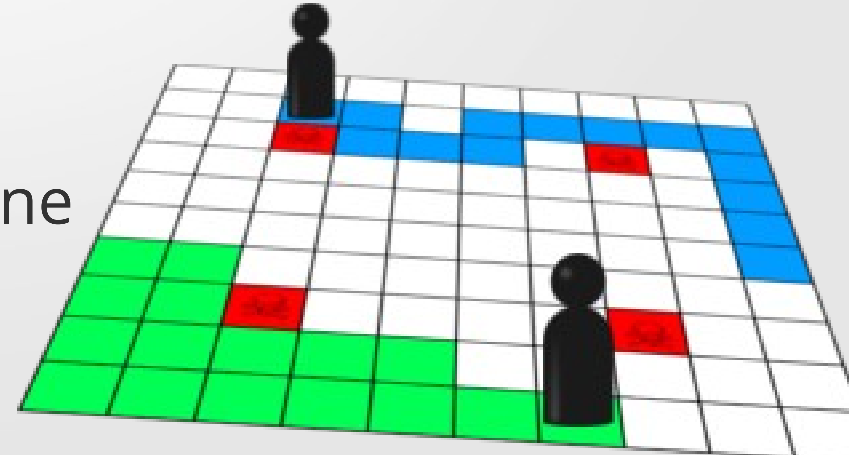
- Change in number of players:
  - If the sensor detects more or less people in the field, the game is paused until solved. A countdown for resuming starts.
- No players are being detected:
  - The game resets after some time.
- No one is staying in the field long enough to signal game start:
  - Idle animation, display some eye-candy, something inviting.

Interaction: Pong game

- In this game, the player's movement on the game surface controls the paddles. The goal is putting a ball behind the adversary.
- The player who scores a set number of points first, wins.
- There is a spring effect of the paddle following the position to improve responsiveness and alleviate jitterring.
- Responsiveness is key for enjoyment of this game.

Interaction: Stepping on Squares Game

- User fills squares and gains territory in the playing field. The only input is running around.
- Round ends when the whole board is full or when the time runs out.
- Red squares are harmful to score.
- Idea: you can't cross your enemy's line
  - Like Tron



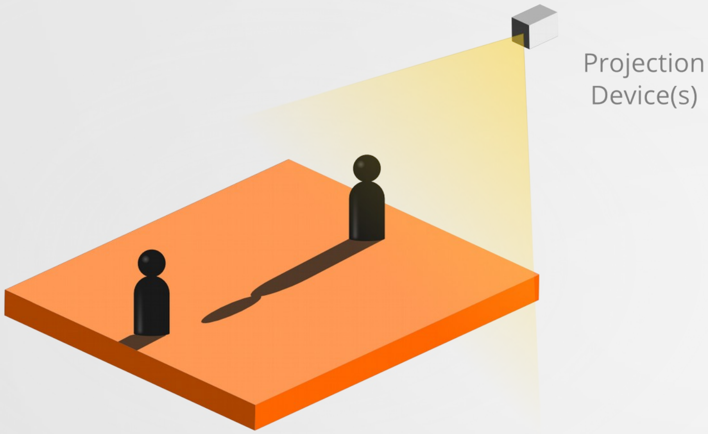
Interaction: Idle mode

- People are passing by the sensor but not stopping.
- Incentivize interaction:
  - Surface displays marker around the user to show that s/he is being tracked.
  - Can be a nice light glow or something more distict.
  - Render nice environment around.
  - Subtle audio cues, e.g. a grave wobble.

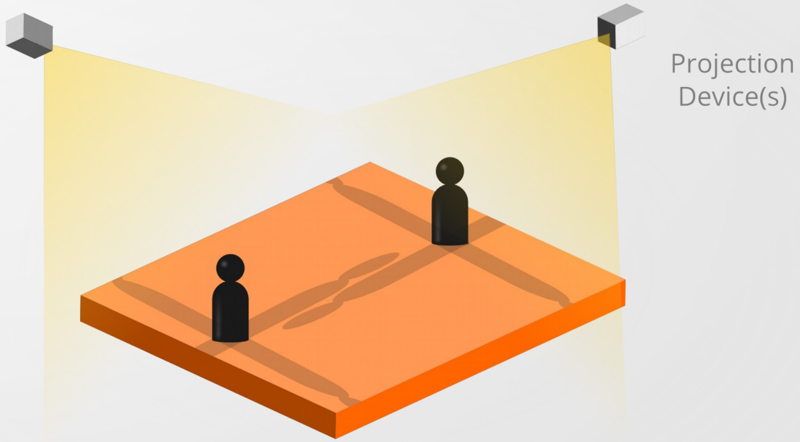
Interaction: Other Games

- Theoretically any game that can be played only with one or more pointing devices can be ported.
- Musical games are a concrete possibility.
  - Two players plays a song stepping on the right notes of a virtual piano on the ground. The best performer wins the musical duel.

Multiple Projection Illustration



Problem: Players cast shadows on the playfield



Mitigation: Use multiple aligned projection devices. Two minimum.

Workload Distribution

- Programming, graphics, testing, evaluation…
- I did everything myself :)
- Estimated work time: 45 hours

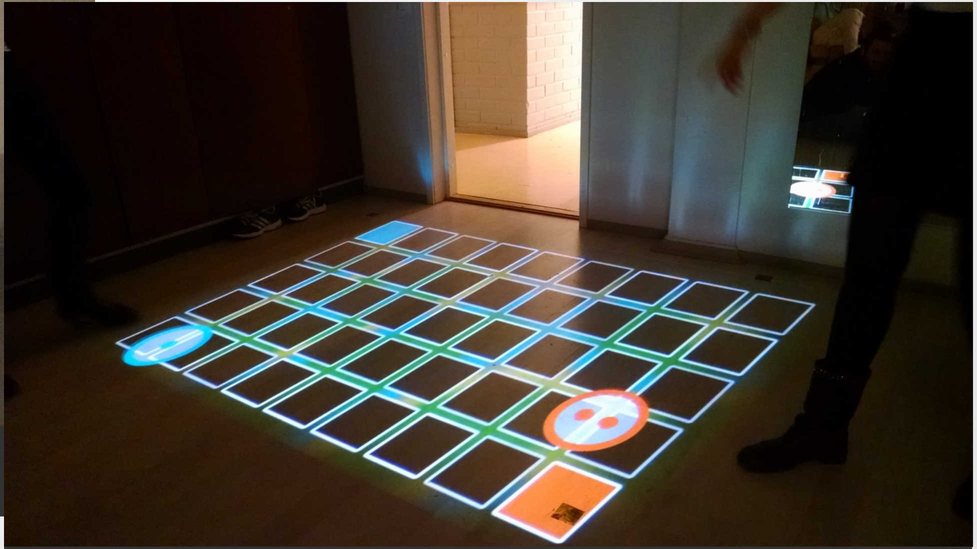
Implementation: Hardware

- For this first implementation I tried to use the easiest hardware to obtain and work with:
  - Core i5 laptop with integrated Intel Graphics and Kubuntu 15.04
  - Standard Kinect from UTA
  - Single projector borrowed from UTA
  - My own table, my own room

PC and Kinect

Implementation: Hardware Setup

Not very safe I guess...

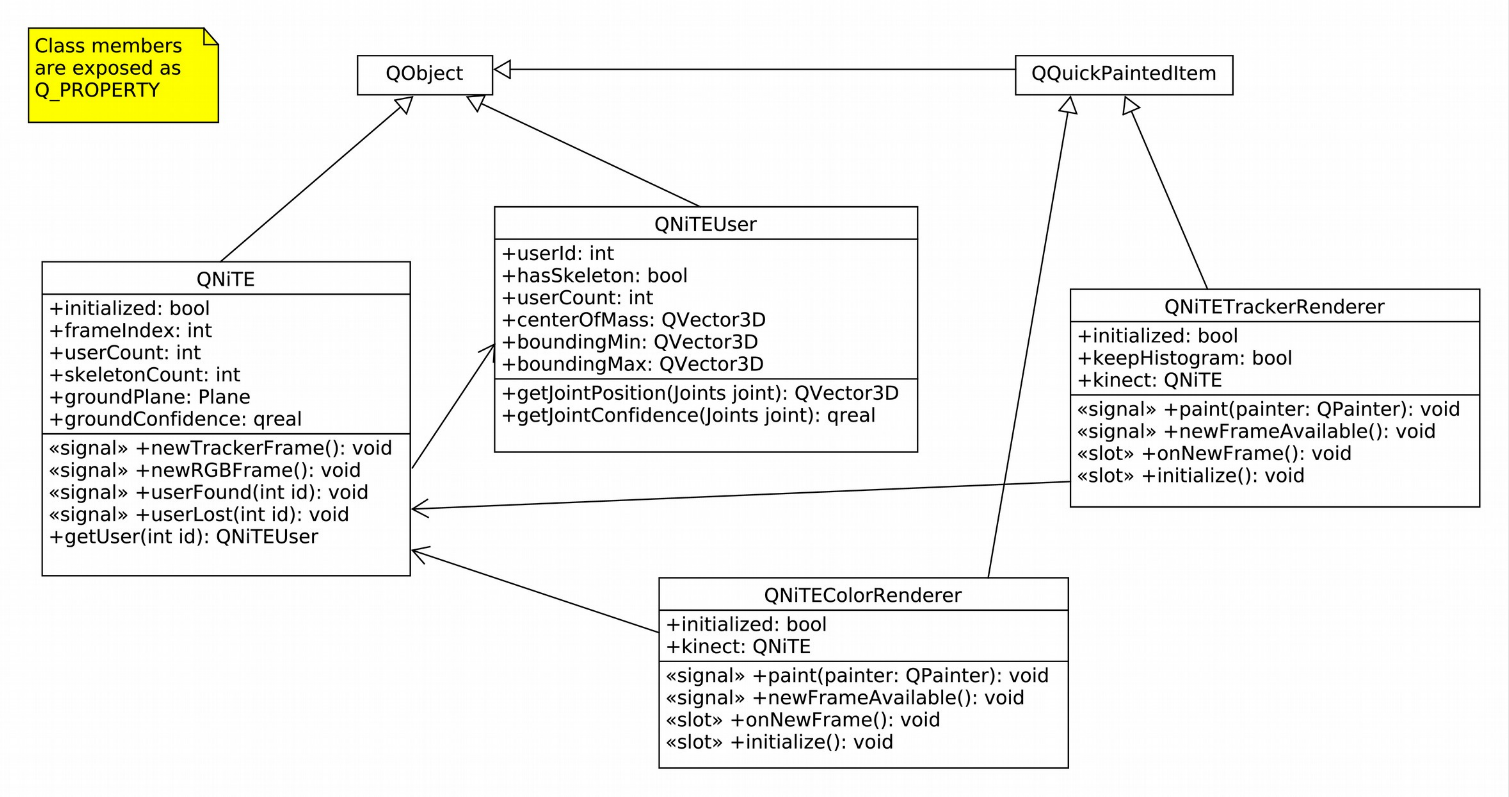


Floor projection test

Implementation: QNiTE

- Made from scratch;
- Wrapper library to access/render cameras and user tracking information from QtQuick;
- Does thread sinchronization and frame processing transparently.
- Data is exposed as Q_PROPERTY members.

QNiTE Class Diagram



Class members are exposed as Q_PROPERTY

QObject

QQuickPaintedItem

QNiTE
+initialized: bool
+frameIndex: int
+userCount: int
+skeletonCount: int
+groundPlane: Plane
+groundConfidence: qreal
«signal» +newTrackerFrame(): void
«signal» +newRGBFrame(): void
«signal» +userFound(int id): void
«signal» +userLost(int id): void
+getUser(int id): QNiTEUser

QNiTEUser
+userId: int
+hasSkeleton: bool
+userCount: int
+centerOfMass: QVector3D
+boundingMin: QVector3D
+boundingMax: QVector3D
+getJointPosition(Joints joint): QVector3D
+getJointConfidence(Joints joint): qreal

QNiTETrackerRenderer
+initialized: bool
+keepHistogram: bool
+kinect: QNiTE
«signal» +paint(painter: QPainter): void
«signal» +newFrameAvailable(): void
«slot» +onNewFrame(): void
«slot» +initialize(): void

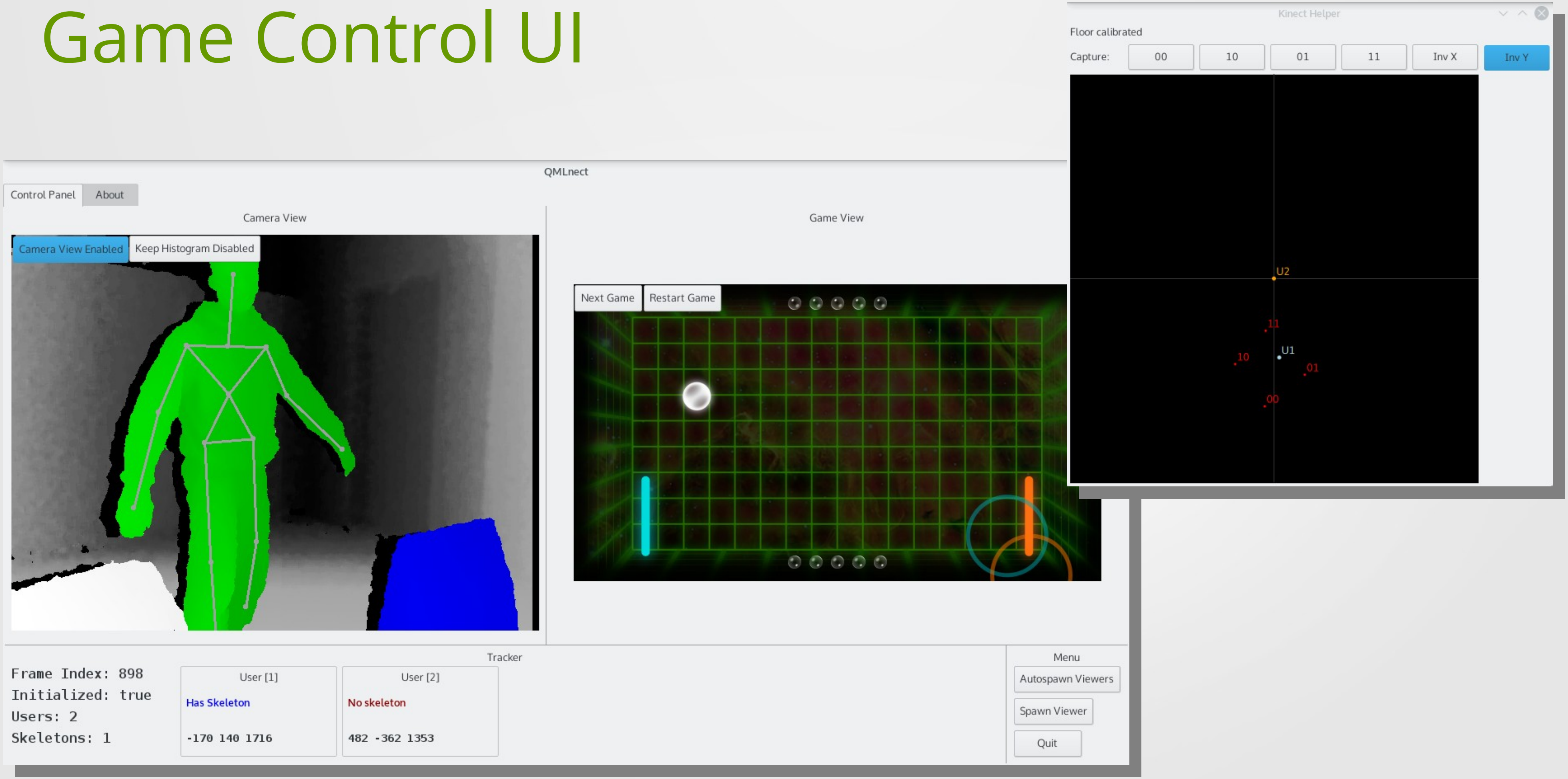
QNiTEColorRenderer
+initialized: bool
+kinect: QNiTE
«signal» +paint(painter: QPainter): void
«signal» +newFrameAvailable(): void
«slot» +onNewFrame(): void
«slot» +initialize(): void

Implementation: neiasound

- A Qt-style library for OpenAL access.
- Facilitates implementation of applications using positional audio.
- I would like other people to give it a try and leave some feedback (2-clause BSD licensed).

  https://bitbucket.org/lpcamargo/neiasound

Game Control UI



Floor calibrated

```
Frame Index: 898
Initialized: true
Users: 2
Skeletons: 1
```

User [1]

Has Skeleton

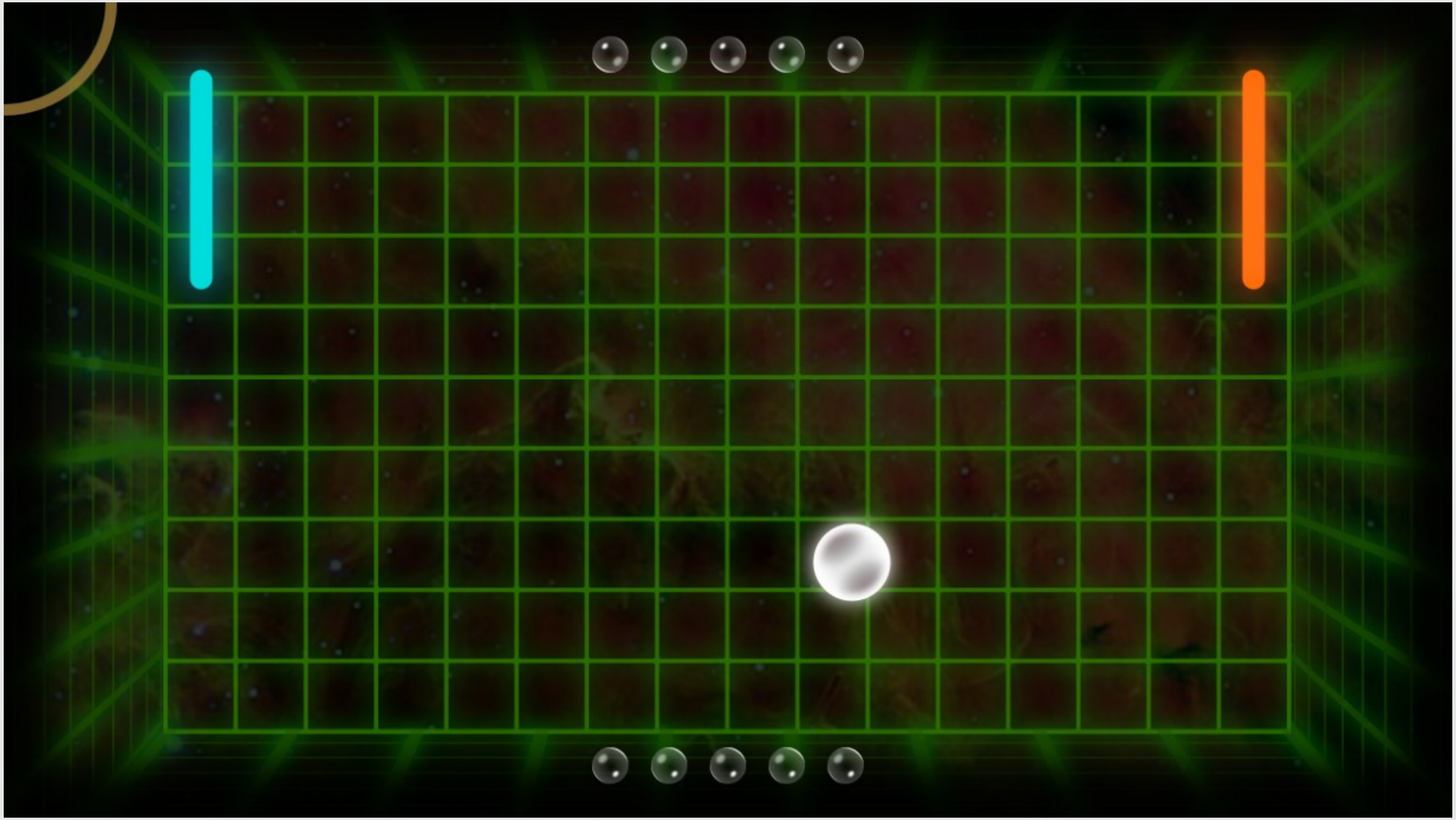
```
-170 140 1716
```

User [2]

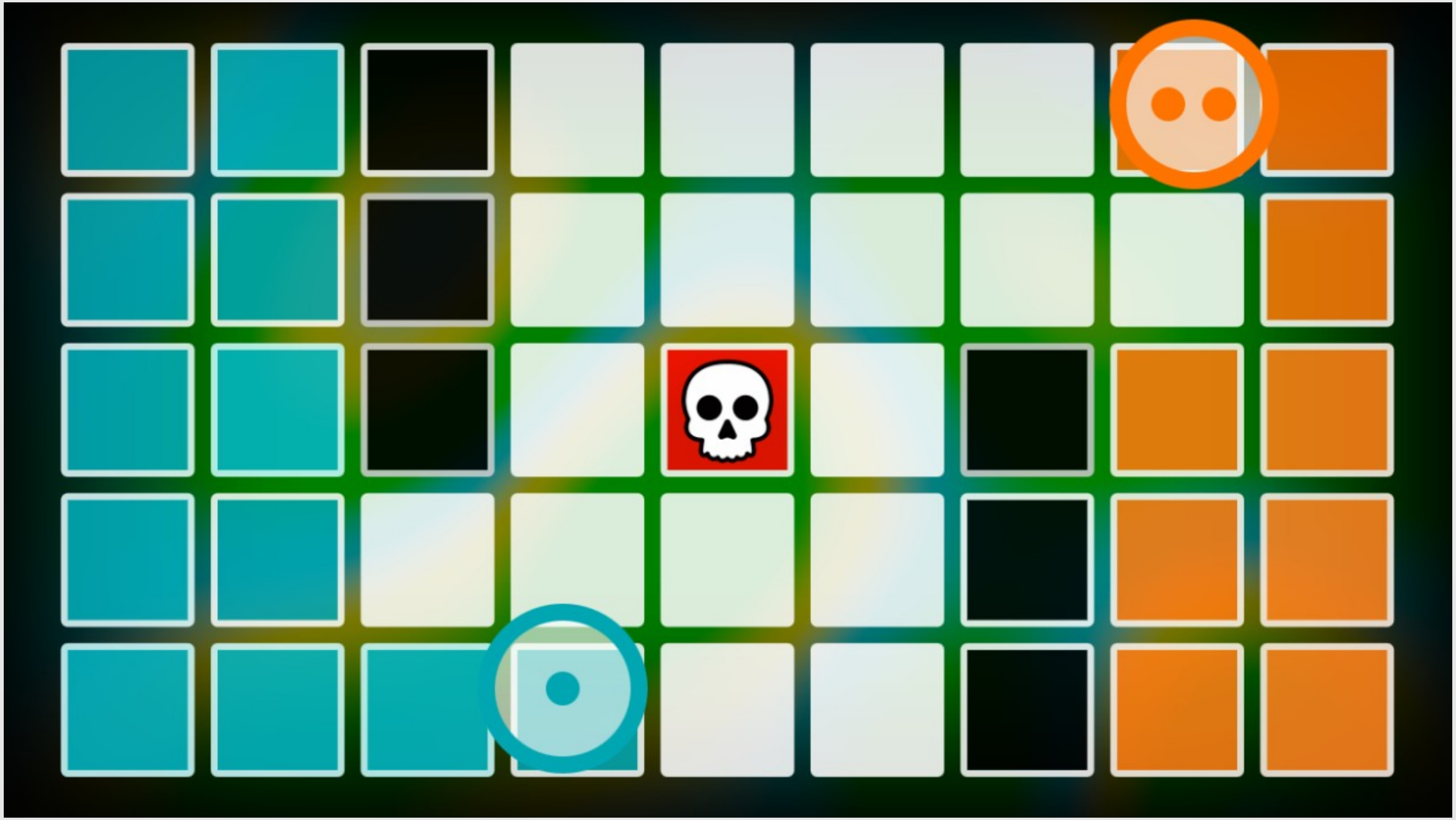
No skeleton

```
482 -362 1353
```

Implemented Game: Pong

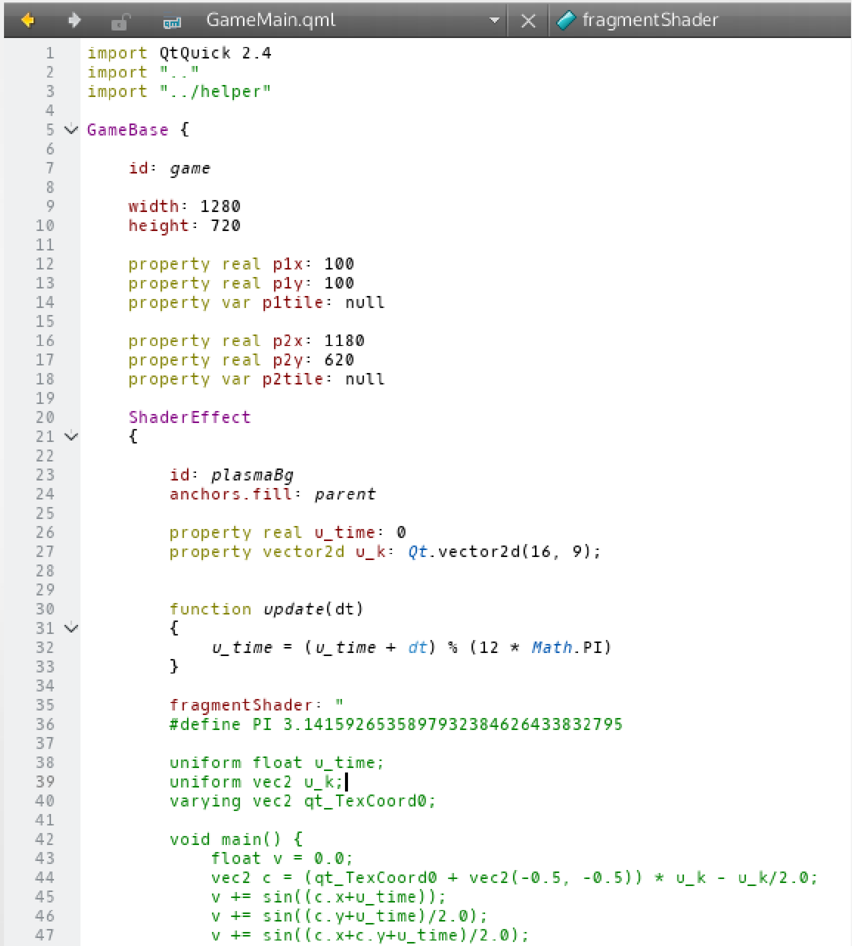


Implemented Game: Squares



Code Statistics

- QML code
  - Floor projection helper: 234 LoC (with UI)
  - Sound helpers: 101LoC
  - Pong game: 331 LoC
  - Grid game: 615 LoC
- C++ code
  - QNiTE: 855 LoC
  - neiasound: 2162 LoC



```
import QtQuick 2.4
import ".."
import "../helper"

GameBase {

    id: game

    width: 1280
    height: 720

    property real p1x: 100
    property real p1y: 100
    property var p1tile: null

    property real p2x: 1180
    property real p2y: 620
    property var p2tile: null

    ShaderEffect
    {

        id: plasmaBg
        anchors.fill: parent

        property real u_time: 0
        property vector2d u_k: Qt.vector2d(16, 9);


        function update(dt)
        {
            u_time = (u_time + dt) % (12 * Math.PI)
        }

        fragmentShader: "
        #define PI 3.1415926535897932384626433832795

        uniform float u_time;
        uniform vec2 u_k;
        varying vec2 qt_TexCoord0;

        void main() {
            float v = 0.0;
            vec2 c = (qt_TexCoord0 + vec2(-0.5, -0.5)) * u_k - u_k/2.0;
            v += sin((c.x+u_time));
            v += sin((c.y+u_time)/2.0);
            v += sin((c.x+c.y+u_time)/2.0);
```

Game Skeleton Example

```
GameBase {
id: game

// Game content goes here

function updatePlayerX(player, x, y) {
// what to do when a player moves around
}

FloorProjectionHelper {
id: helper
}

}
```

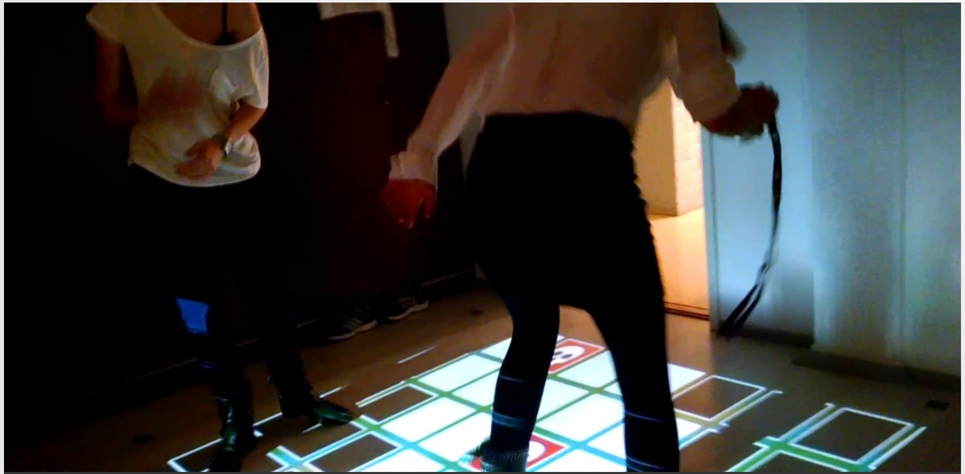
- A game for the system is no different than any QML game.
- But instead of using normal input code, the game uses a `FloorProjectionHelper` component instance.
- It can also read the user's skeletons if desired.

User Evaluation: Test Users

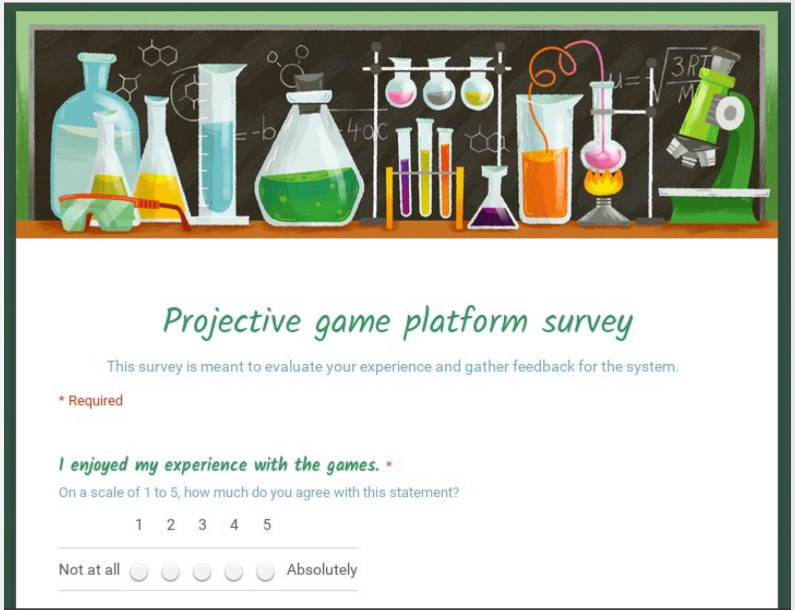
- Exchange students from my building.
- A mix of friends and people I don't really know.
- I was just grabbing them by the corridor.
- 8 people answered the survey, more people tried it out.
- After the evaluation a small party formed around the games!

User Evaluation: Procedure

- Users tried both games in pairs.
- Later, they were asked to fill in a short questionnaire.



Evaluation Session



Projective game platform survey

This survey is meant to evaluate your experience and gather feedback for the system.

* Required

I enjoyed my experience with the games. *

On a scale of 1 to 5, how much do you agree with this statement?

	1	2	3	4	5	
Not at all	○	○	○	○	○	Absolutely

Questionnaire (Google Forms)

User Evaluation: Methodology

- The questionnaire contained scale-type evaluation questions. Eg.:
  - I enjoyed my experience with the games.
    - On a scale of 1 to 5, how much do you agree with this statement?
- It also had fields for open critique and suggestions.
  - Would you like to suggest some changes to the experience?
  - Any other comments?

User Evaluation: Results

- I enjoyed my experience with the games: 4.75 (out of 5)
- I felt in control of the system at all times: 3.125 (out of 5)
- The games were intuitive to play: 4.75 (out of 5)
- Graphics, sound, feel, were satisfactory: 4.625 (out of 5)
- Likeliness of intereaction in a public space: 4.25 (out of 5)

User Evaluation: Results

- User feedback was generally positive. Everybody had fun.
- The least identified aspect was player control.
- It can be a bit disorienting at times.
- It was noted that the presence of friends improve the experience dramatically.
- One of the players ocluding the other from the kinect was a problem.
- Better camera positioning and application of more projectors are likely to improve the experience

Thank you!