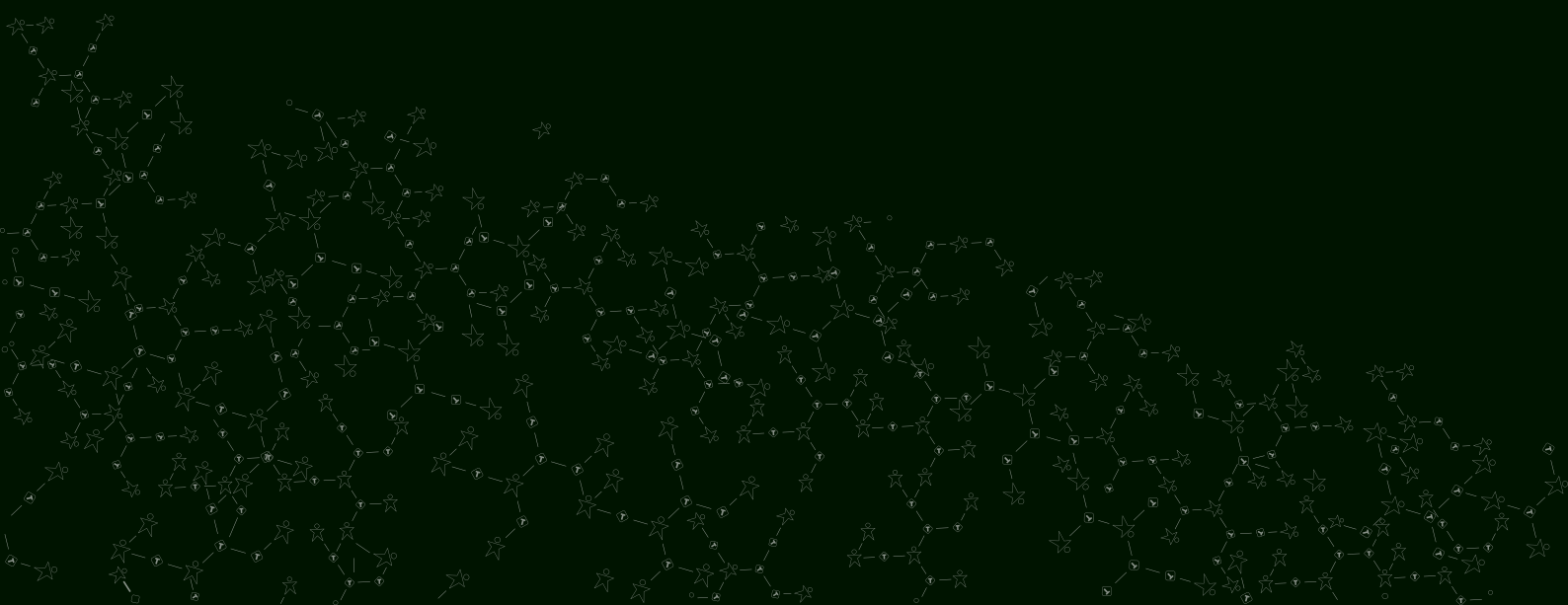
Growing Systems

Making the Virtual City Accessible in the Physical

Concept: SID (Showcase Industrial Design)



Growing Systems:
Making the Virtual City Accessible in the Physical

DPI42

Theme: Out of Control

Project by Lars Hottentot S108661

Coach: M.S.A. de Koning

June, 2013

Abstract

The focus of this project was to create a system that connects a growing virtual data set with a physical locus of interaction, in the context of the city. Initial research and brainstorming focused around the aspects of a city and its digital and analog interactions. As a result, it was found that the university could also be considered as a part of the city. The final concept aims to solve a problem within the Industrial Design department at the Technical University Eindhoven. The problem is that there is no database of finished projects by students, and therefore no method of seeing what has been done before within the study. The final concept solves this by proposing a physical kiosk that allows visitors and students to view the previous projects with an added layer of augmented reality, giving users a chance to view the prototypes in 3D virtual/physical space and navigate easily through the entire database.

INDEX

INTRODUCTION

From the official project description.

"Currently the city around us is coming to life in the digital world. How this digital city becomes meaningful to us remains to be seen but the first signs point towards visual solutions like augmented reality (e.g. layar) or SMS-messages. Imagine and go beyond scenarios like a cinema that contacts you with a deal on the last to tickets to the movie that is about to start when you walk by, or a grocery store that advertises their vegetables that are about to expire. While these examples illustrate part of this project, the project is not just about location dependent advertisement or location based services per se. You are encouraged to find new areas for this system to grow in, within the limits of the design challenge.

One of the ways to approach it is for example interactive public art installations. The current development in digital public arts involves a significant amount of new carriers in not only material, but also in technology, resulting new dynamic and interactive forms that require the artists and designers to construct their work from a system view and with a good understanding of human-system interaction and related interface technologies. It is no longer about carving stones and casting bronze; it is time to sculpture the interactive experience."

The design challenge is to design a physical locus of interaction that allows for digital actions, connecting the virtual city to the physical. The idea is to let this system change and grow over time and in specific situations, allowing for an ever growing list of possible interactions. The project should focus on meaningful interaction between these two worlds; the physical and the virtual.

Fig 5.1 Infoboard that the public can post on to share messages. Foto: Kim Sauvé

RESEARCH

City Research

The team decided that the best initial step was to go out into the city and observe. This gave insight into current systems involving the digital world and the city and revealed possible situations and locations that could be investigated further. A 6 minute video of this excursion can be found HERE.
Besides a few obvious examples such as the DropStuff board where you can drop videos and messages to share with everyone to see publically, as shown in figure 5.1 on the previous page, there did not seem to be many examples of connections between the virtual and physical city, much less an interface to control such a system.

BrainStorms

Initial brain storms around the individual topics of the city and digital activities revealed the breadth of options available (figures 7.1 and 7.2). Realizing the city is a combination of the physical and the social, another brainstorm was held to help define what the social aspects are of a traditional, analog city, and what physical locations or situations could be designed for (see figure 8.1).

A further development of this last brainstorm used a 50-5-1 method combined with random keyword association. Many different keywords, originating from the first three brainstorms, such as location or situation, were written on pieces of paper and drawn blindly from a bowl. This forced the team to think outside of the box and sometime create some very strange combinations. After about fifty of these were made, the best ones were selected to work out further. Some of these results can be found on page 8.

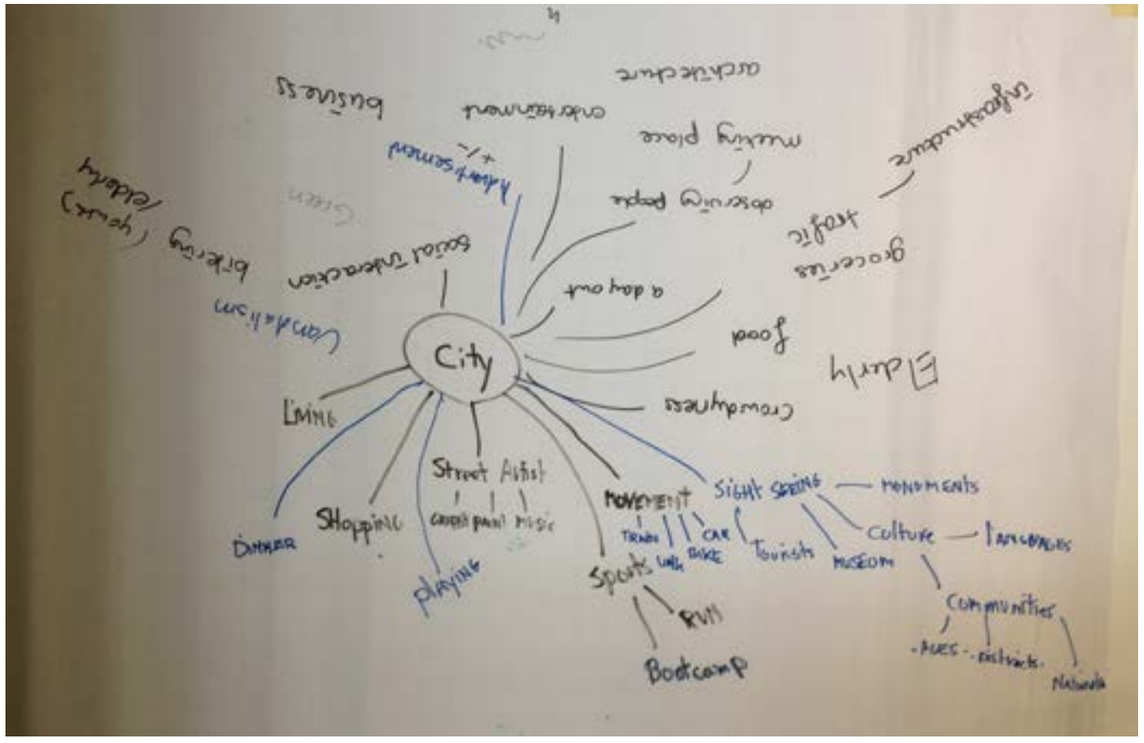


Fig 7.1 Brainstorm on what a city consists of.

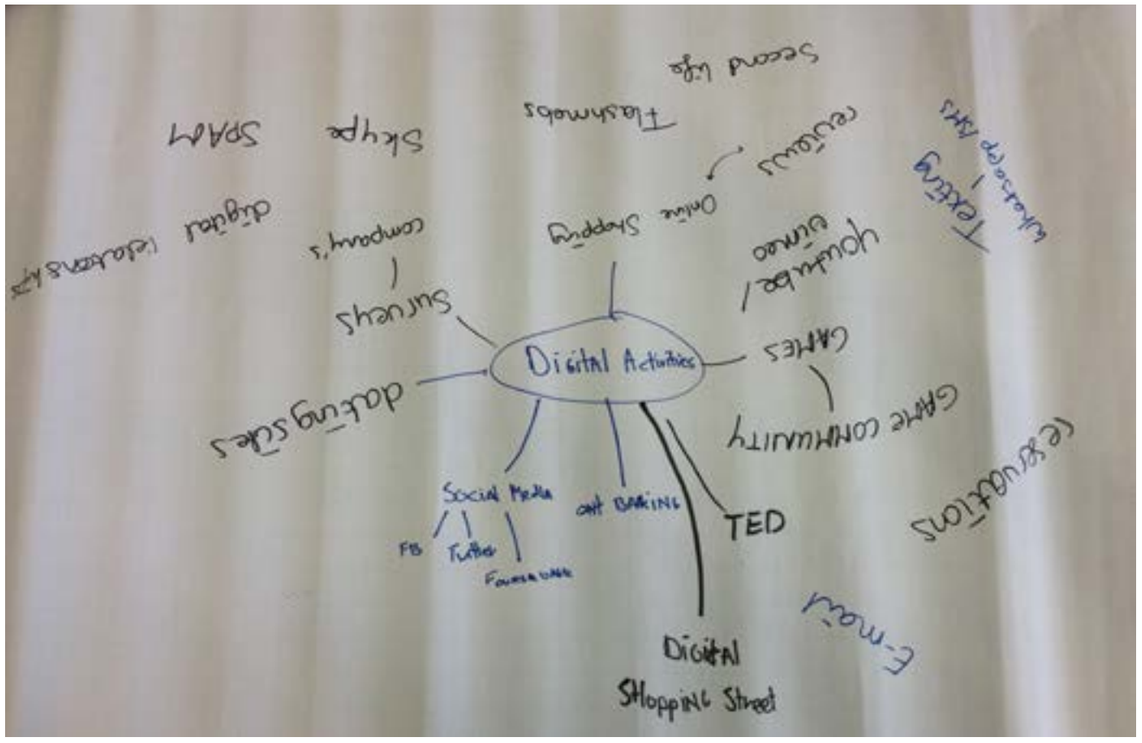


Fig 7.2 Brainstorm on digital activities and services.

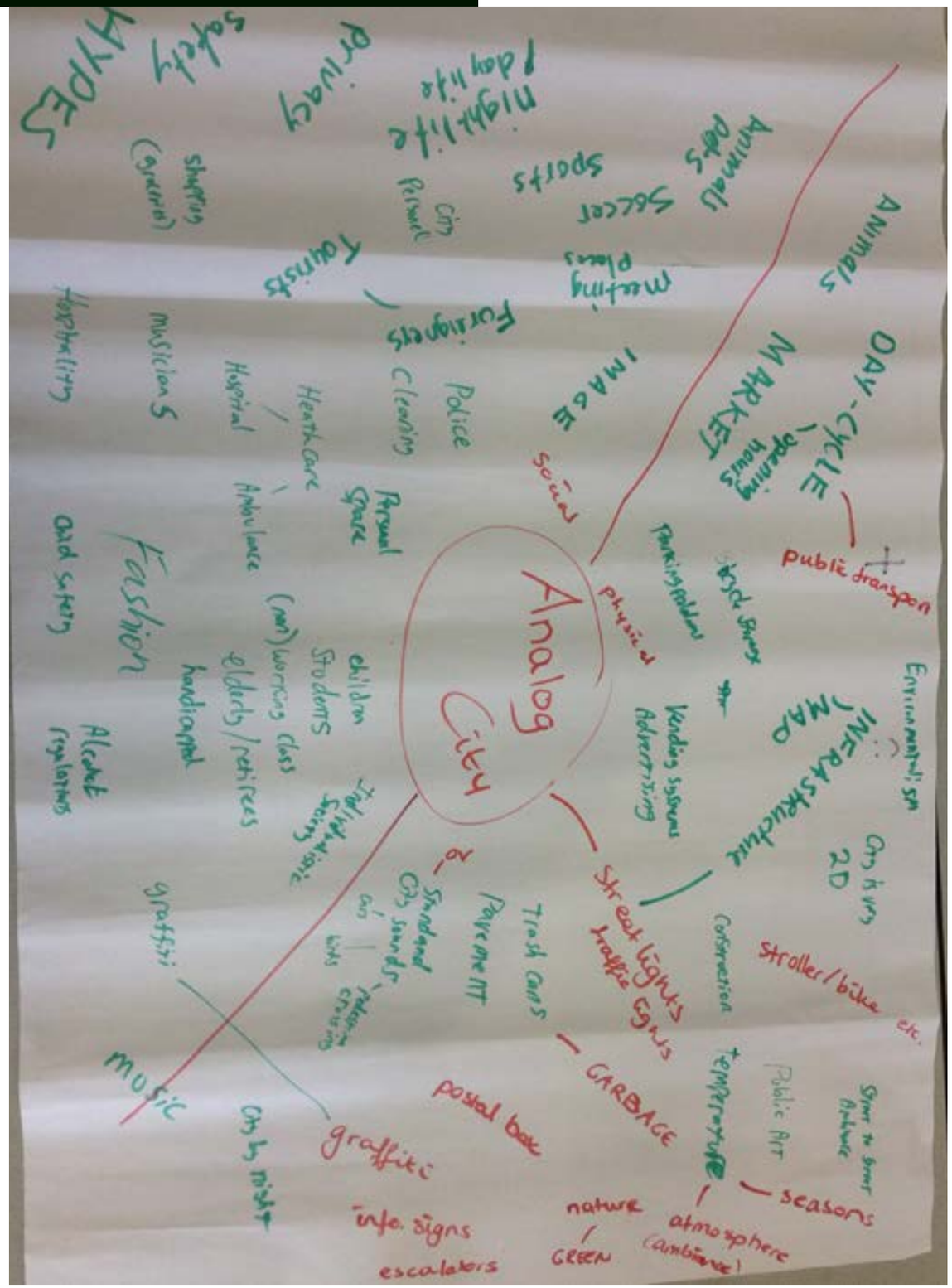


Fig 8.1 Brainstorm on the analog city, split between social and physical.

Brain Storm 50-5-1 Results (Part Selection)

Traffic Lights + Graffiti + Texting

- Put text inside red light for a user to read, keeping their attention on the red light.
- Use texting to influence the messages inside the lights. Maybe combine it with speeding warnings.

Traffic Lights + Fireworks + Texting

- Exaggerated red light, lighting up the entire road below
- Traffic light knows if you are using your phone, and launches fireworks or something similar to get your attention on the road again.

Map + Candy

- Create a virtual candy trail. A person searches for a specific candy trail that has been set by other players, letting players guide each other to places through a mini adventure.

Aging + Market + Youtube

- Supermarket with trees so you can pick your own fruit, combined with online streaming so you can see if the store still has the fruit you want. (Not including aging as a criteria).
- Document the wine process of each batch of wine made and let the users see how their wines were created through video.

Social + Bikes

- Networked bikes that connect to other bikes going a similar speed, and allow you to talk to strangers whilst traveling.
- Bikes that suggest new routes to bike to your destination, giving you more freedom to explore new parts of the city.

Children + Vandalism

- A growing object for vandalism. Give the youth an incentive to vandalize a specific object to prevent vandalizing other objects. The object takes the punches and graffiti and changes accordingly.

Mobile + Running + Map

- A running application that notifies you of traffic light status before you get there, giving a so called “green wave”

CONCEPT

Concept Selection

The brainstorms resulted in some interesting concepts, some of which can be seen on the opposite page. However, as these concepts were listed, it became clear that some of these ideas may have been worked out before within Industrial Design Eindhoven. Up until then, the team had been looking at existing situations and virtual data within the context of the city, but what if the virtual data still had to be created? During an initial brainstorm, the idea of Augmented Reality came up. This combination of a virtual overlay over the physical world epitomizes the essence of the project. Combined with the realization of a lack of database for Industrial Design, the project took an unexpected turn. The result was to create an augmented reality platform for Industrial Design, coupled to a database of projects with 3D object files attached, so one can see the product in 3D space. The location of such a system would be the Technical University Eindhoven's main building hallway, where visitors and students alike can approach the platform and browse through all the projects of Industrial Design, sharing with the world the projects of Industrial Design.

Theme Day

At the theme day halfway through the semester, a video was shown. It can be found here. This video depicted a futuristic look at what the concept could look like, with gesture based controls and a translucent information screen (see fig 11.1). The video showed a goggle-less way of viewing augmented reality, but society is many years from that.
The feedback on the system during this theme day was positive overall, but there was advice to investigate how to connect physical interaction into the system, as it appeared it might as well be a website with a mobile application attached. Thus, methods of physical interaction were needed to create more of a connection between the digital and physical worlds.

Fig 11.1 Screenshot from video, showing futuristic aspects of the concept.

DEVELOPMENT

Defining the Project

With a concrete idea in mind, the next step was to define exactly what the goal was for the rest of the project.

"To create a permanent virtual/augmented reality exhibition to showcase Industrial Design projects. This means creating a virtual model of the product to be viewed through augmented reality technology. The system will have a growing database of projects to be showcased, and allow the observer to partake in a rating system. The rating system can be done at a direct level, such as giving a number of stars to a project or switching the ranking of any two , or at an indirect level, by basing it on time spent looking at each project."

As realizing the entire project would not be possible due to the time limit and lack of knowledge, it was decided to focus on getting the augmented reality part of the concept working, with the database and rating system to be left for future development, but still fully defined.

The process started with researching current methods of augmented reality. There are two main methods. One involves using triangulation by measuring the orientation of the viewer's glasses/tablet from an externally placed camera. This system then updates the screen of the viewer to correspond to the angle. The other method is by using a camera as the perspective, and looking for markers in that image that correspond to a stored image. If the system is able to place the image, in 3D, onto the markers, then it knows what the orientation of that marker is, and where it is. This allows an entire world to be created digitally using only that single marker as a reference.

As the system being created is small scale, there is no need for external cameras, and a single marker should be enough to overlay a virtual object with the physical world. For simplicity, it was decided to use a tablet device with a camera to simulate the viewpoint. This tablet could then be picked up and moved around the displayed marker, introducing part of the physical interaction suggested at the theme day.

Fig 13.1 Sketch of what the end system might look like.

DEVELOPMENT

Coding

Whilst looking for a method of creating this augmented reality environment, a simple example was found that introduced augmented reality by reading a picture with markers using the application Processing, a java based program. This was a good start, but now live video had to be introduced instead of a picture. After some more research, a library was found that could be used to import video. A Library is a like a reference book; in the main code, you can write a single line of code that references to a few pages of code in the library, allowing for simpler code.

After introducing the video into the code, it was noticed that the frame capture rate of the camera was extremely low (near 1fps). After some troubleshooting of a few days, the capture resolution was set much lower, and as a result it worked near real-time. However, this reduction in resolution also meant a loss of accuracy, and markers needed to take up more of the screen before being recognized. As this was currently unsolvable, it was left as is.

Then came the last part of the code: importing 3D objects. As there was no example on how to integrate 3D models into augmented reality, there was a lot of trial and error for weeks. A library to import .obj files was found, but there were a lot of compatibility issues. As a result, it was impossible to export the code as a stand-alone application to a tablet, which was the initial idea. However, the code did work on pc within the processing application, so it was decided to use the tablet as an extended desktop and run it on the tablet's screen. This would prove to be a better method, as now it would be directly connected to the rest of the system comprising of the database and information screen, allowing for quicker updates.

Effort was also put into trying to use the tablet's built in camera, but to no avail. Therefore, an external webcam was used and a casing was built to house the units together.

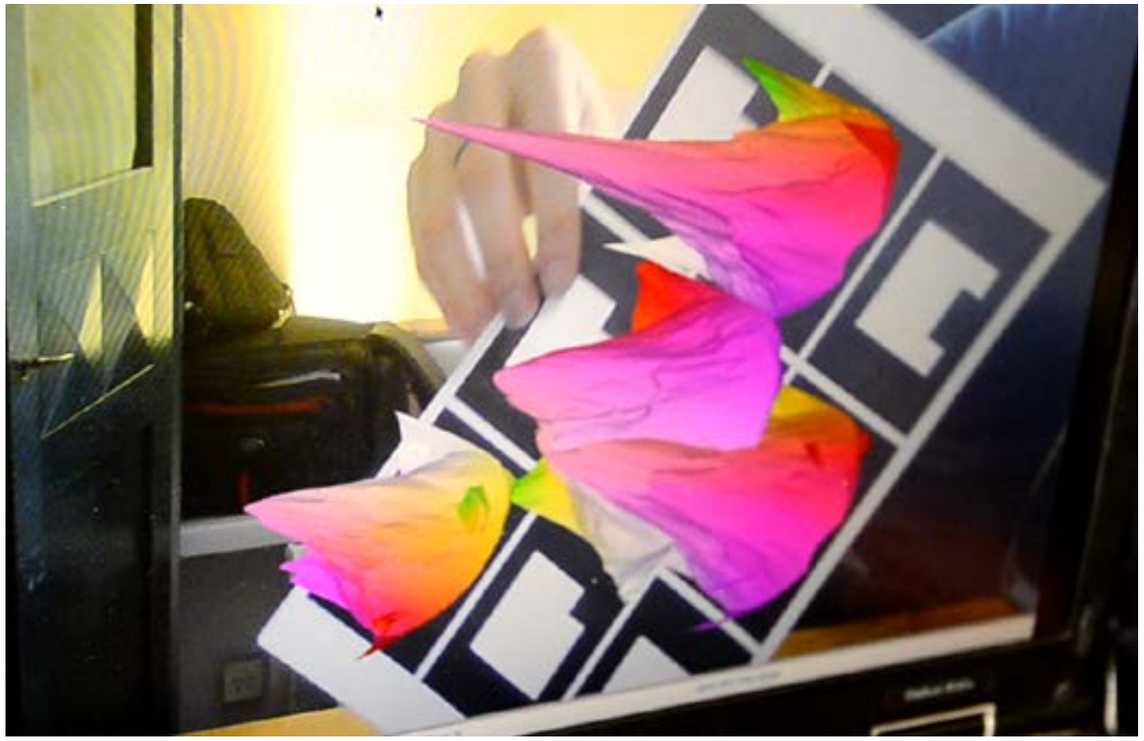


Fig 15.1 Initial success with reading markers and mathmatically generated mountains

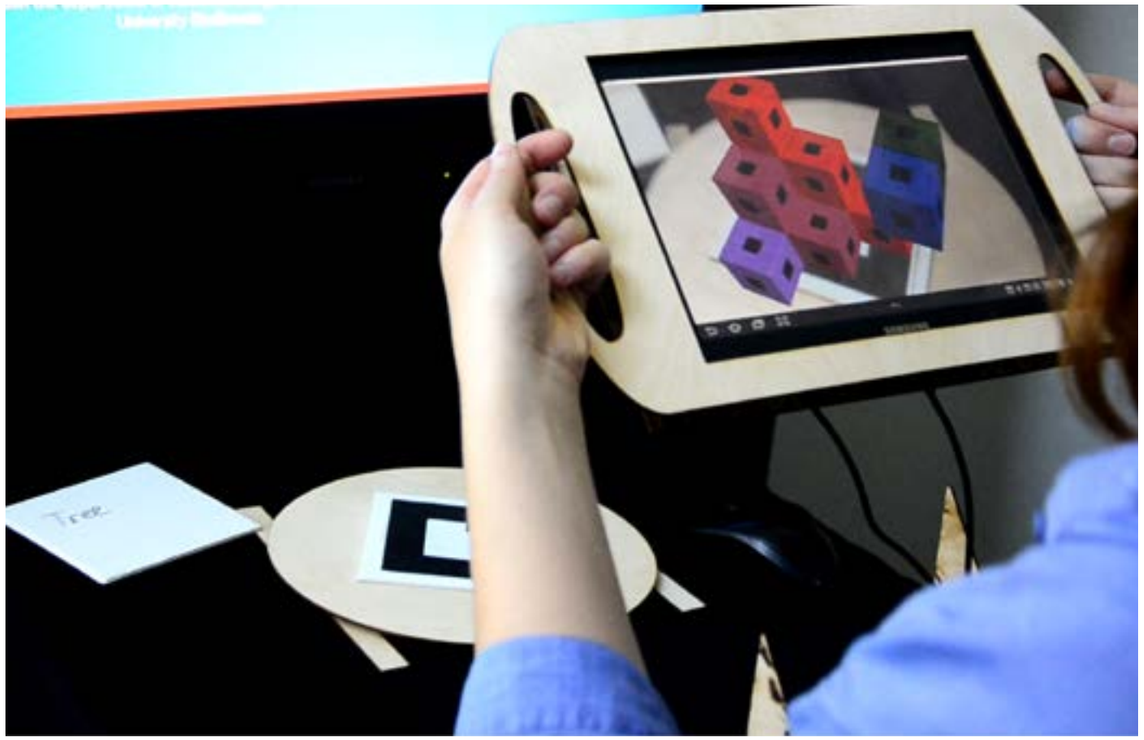


Fig 15.2 Succes with importing the object files into the program and rendering on screen

DEVELOPMENT

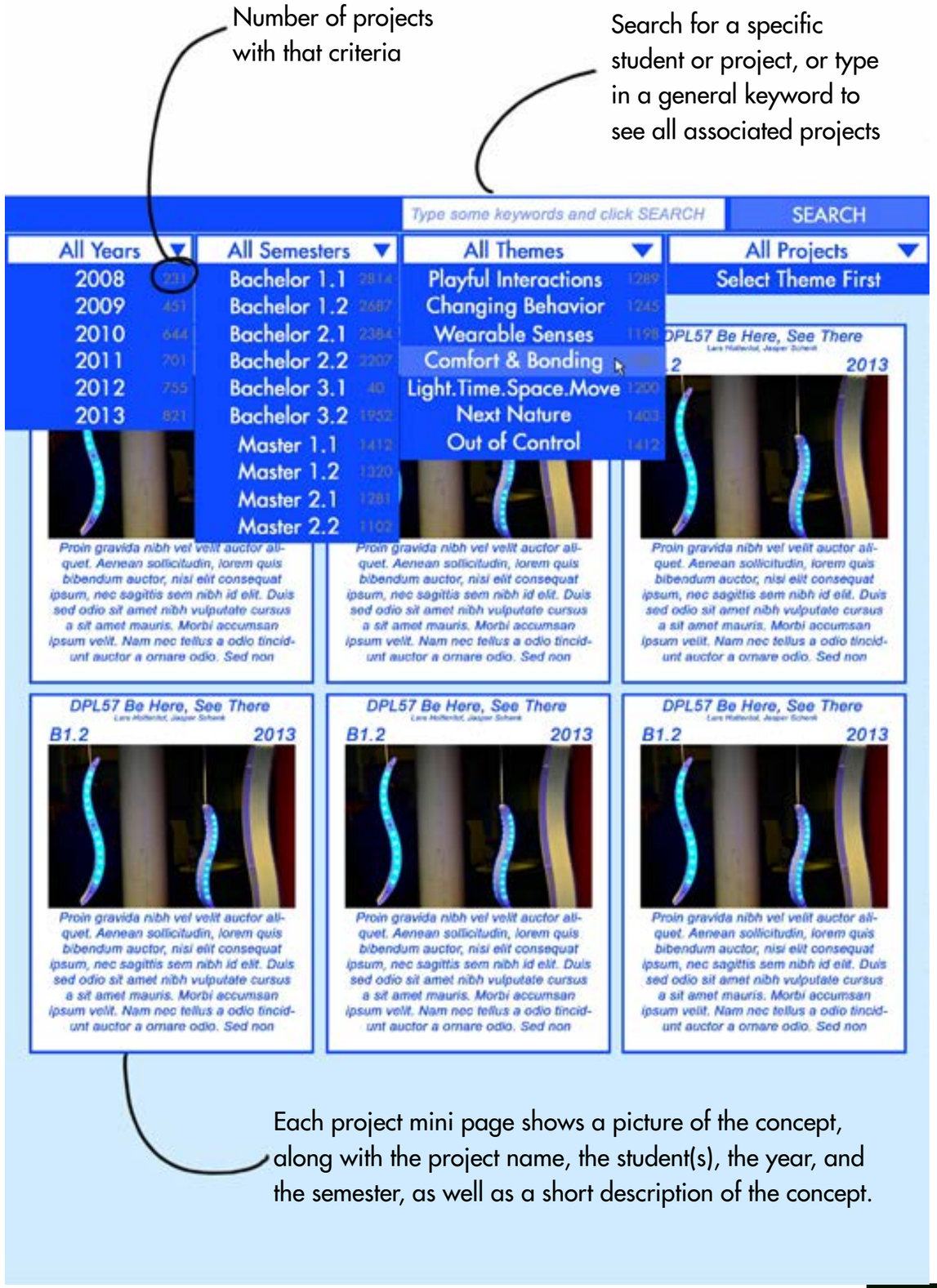
Database Structure

The database should be created in such a way that it is easy to navigate, efficient in displaying results, and have a low threshold of learning, so that users can understand immediately how it works. After researching possible database systems and navigation options by visiting website with databases, there appeared to be only one that seemed fitting to the situation. The opposite page shows how this database should work.

The idea is that each project page has its own miniature version, as shown. These mini pages have certain keywords attached to them, such as their year, semester, theme, project name and other metadata such as the student who created it. The criteria selector on top allows the user to select which criteria to filter on. Selecting a combination of criteria, for example "2010" and "Playful Interactions", would result in showing only those projects done in that Theme and Year. Furthermore, a user can use the search feature to look for particular keyword such as "Communication" to display all related projects, or type in a specific student's name to find all the projects done by him or her.

The grey numbers next to the selectors show how many projects there are in that specific criteria. However, it extends to the other selectors as well. For instance, if you select the theme Playful Interactions, you will see 1289 next to it, indicating the number of projects with that keyword, but if you have already selected to only see the year 2010, the number next to Playful Interactions will update to show how many projects were done in only the year 2010. Each criteria selected will alter the number of projects with the next criteria selection. The list of 2008 Bachelor 2.1 Playful Interaction projects would be a lot lower than any single criteria, allowing the user to search very specifically and minimizing the need for scrolling through endless pages of links.

Another feature to be implemented, but not shown yet, is the sort function, where one can sort the project according to project code, student name, or even popularity. The popularity function would work by registering the amount of times a project has been viewed.



PROTOTYPE

Final Experiential Prototype

Presenting SID: Showcase Industrial Design.

The final experiential prototype shows the capability of augmented reality, and proposes that the entire department of industrial design be documented in both old media (website/text) and new media (virtual experience). The casing was designed to hold the tablet and webcam together, and invite users to grab the unit. The database will grow with each semester and provide the department of Industrial Design a medium to distribute its ideas beyond the university.

The process starts with the project report, which is attached to the project page within the database as a link, which users can send to themselves via email if they want more in depth detail on the project. A brief description is added to the page, along with photos and video, and an even shorter description that fits on the mini page. The prototype will then be scanned into a computer to create a 3D file, which can then be linked to a marker. When the user browses through the database and selects a project, that project is displayed on the main screen. At the same time, the second screen on the table updates to show the marker. The user can physically rotate this screen through means of a Lazy Susan, adding another layer of physical interaction to the system. Currently this second screen has not been realized, and the system uses physical prints of the marker that need to be switched out, but this was due to costs. After that, the camera picks up the marker, and projects the 3D file onto the screen.

Fig 19.1 Final exhibition set up, with the augmented reality system and the lazy susan marker holder.

FUTURE

First and foremost, this prototype is only part of a larger system. As this project focused on just the experiential interface, there is still more to be created. First, the system needs to become more stable, as it was very CPU intensive. Then the marker screen needs to be integrated into the system and the entire system needs to be designed with form and senses in mind, as currently it is all about the technology. Finally, the entire database structure needs to be created and projects need to start being documented. The final result will be a kiosk like package that is robust, independent, and eye catching. The department of Industrial Design can even bring this kiosk to external events to promote the department, and perhaps give companies a chance to contact students about certain projects they want to invest in.

Overall, the proposed system, if implemented, will give students a chance to see what has already been done, give visitors an impression of what the students at Industrial Design do, and make sure developments over the years are not lost, which can be considered an ideal platform for the future development of the department.

For more information, please contact
Lars Hottentot
at

L.hottentot@student.tue.nl

APPENDIX

CODE for Processing Program PART 1

```
//The following code is a combination of different examples.
//
//1. Example with reading an image to overlay augmented reality objects.
//http://www.creativeapplications.net/processing/augmented-reality-with-processing-tutorial-processing/

//2.Saito ObjLoader for the import of 3d models.
//http://www.processing.org/discourse/beta/num_1215851181.html
//
//3. using GSVideo to import webcam feed, part of example 1.

float rotX, rotY;

import java.io.*;                          // for the loadPatternFilenames() function
import processing.opengl.*;                // for OPENGL rendering
import jp.nyatla.nyar4psg.*;               // the NyARToolkit Processing library
import codeanticode.gsvideo.*;             // the GSVideo library
import saito.objloader.*;                  //import the object loader library

//define the names of the different object models
OBJModel ft1 ;
OBJModel ft2 ;
OBJModel ft3 ;

MultiMarker nya;                           //define Multimarker as "nya"
GSCapture cam;                             //define camera as "cam"

// this is the arraylist that holds all the objects
ArrayList <ARObject> ars = new ArrayList <ARObject> ();


// the full path to the camera_para.dat file, where the camera parameters are stored.
String camPara = "D:/User Files/My Documents/Processing/Processing 1.5.1/processing-1.5.1/lib/NyAR4psg/data/camera_para.dat";
// the full path to the .patt pattern files
String patternPath = "D:/User Files/My Documents/Processing/Processing 1.5.1/processing-1.5.1/lib/NyAR4psg/patternMaker/examples/ARToolKit_Patterns";
// the dimensions at which the AR will take place.
int arWidth = 640;
int arHeight = 360;
// the number of pattern markers (from the complete list of .patt files) that will be detected, here the first 10 from the list, even though we are only using 3
int numMarkers = 10;


void setup() {
  size(1280, 960, OPENGL); // Setting the size of the display screen

  String[] cameras = GSCapture.list(); // detect all the cameras and make a list

  if (cameras.length == 0)  // if there are no cameras detected
  {
    println("There are no cameras available for capture.");
    exit(); //no cameras, then quit
  } else {
    println("Available cameras:");
    for (int i = 0; i < cameras.length; i++) {
//if cameras are found, then give them numbers starting with 0
      println(cameras[i]);
    }
  }
  cam = new GSCapture(this, 640, 480, cameras[0]); // initialize the webcam capture at that resolution, selecting camera 0 from the list
  cam.start(); // start capturing
  // initialize the MultiMarker at a specific resolution
  nya = new MultiMarker(this, arWidth, arHeight, camPara, NyAR4PsgConfig.CONFIG_DEFAULT);
  // set the delay after which a lost marker is no longer displayed in milliseconds
  nya.setLostDelay(1);
  // load the pattern filenames (markers)
  String[] patterns = loadPatternFilenames(patternPath);
  // for the selected number of markers...
  for (int i=0; i<numMarkers; i++) {
    // add the marker for detection
    nya.addARMarker(patternPath + "/" + patterns[i], 80);
    // and create an ARObject with the corresponding 'ID'
    ars.add(new ARObject(i));
  }

    stroke(255);
    noStroke();

 ft1 = new OBJModel(this, "ProjectTree.obj", "absolute", TRIANGLES); //define model ft1 and point it to the location of the .obj file
    ft1.enableDebug(); //debug allows for monitoring in console for errors

    ft1.translateToCenter(); //move the object to the center of the AR

 ft2 = new OBJModel(this, "Sleepaid2.obj", "absolute", TRIANGLES);
    ft2.enableDebug();

    ft2.translateToCenter();

 ft3 = new OBJModel(this, "musiccubes.obj", "absolute", TRIANGLES);
    ft3.enableDebug();

    ft3.translateToCenter();
}

void draw() {
  // if there is a cam image coming in...
  if (cam.available()) {
    cam.read(); // read the cam image
    background(0); // a background call is needed for correct display of the marker results
    image(cam, 0, 0, width, height); // display the image at the width and height of the sketch window
    // create a copy of the cam image at the resolution of the AR detection (otherwise nya.detect will throw an assertion error!)
    PImage cSmall = cam.get();
    cSmall.resize(arWidth, arHeight);
    nya.detect(cSmall); // detect markers in the image
    // set the AR perspective uniformly, this general point-of-view is the same for all markers
    nya.setARPerspective();
    // run all the ARObjects's in the arraylist => most things are handled inside the ARObject part of the sketch, which is a different file
    for (ARObject ar : ars) { ar.run(); }
    // reset to the default perspective
    perspective();
  }
```

```
}
  void mouseDragged() // allow for the mouse to move the objects digitally as well
{
    rotX += (mouseX - pmouseX) * 0.005;
    rotY -= (mouseY - pmouseY) * 0.005;
}

// this function loads .patt filenames into a list of Strings based on a full path to a directory (relies
on java.io)
String[] loadPatternFilenames(String path) {
  File folder = new File(path);
  FilenameFilter pattFilter = new FilenameFilter() {
    public boolean accept(File dir, String name) {
      return name.toLowerCase().endsWith(".patt");
    }
  };
  return folder.list(pattFilter);
}
```

Code PART 2 (ARObject reference within first part of code)

```
// class that defines the ARObject, both the AR detection and display are handled inside this class
class ARObject {
  int ID; // keep track of the current the ID of the object (corresponds with the ID i of the marker)

  ARObject(int ID) {
    this.ID = ID; // set the ID for the markers
    }
  void run() {
// if the marker is found, go to display code
    if (nya.isExistMarker(ID)) { display(); }
  }
  void display () {
    // get the Matrix for this marker and use it (through setMatrix)
    setMatrix(nya.getMarkerMatrix(ID));

    // Create lights to show shading in models
     directionalLight(126, 112, 126, 1, 1, -1);
    ambientLight(126, 102, 126);

    // Set up rotation through mouse click in augmented
    rotateX(rotY);
    rotateY(rotX);

  // if marker 1 is found
  if (ID == 5) {
              // turn things upside down to work intuitively for Processing users
              scale(1, -1);
              // hover the cube a little above the real-world marker image
              translate(0, 0, 0);
              //scale the size of the object to correct display size
              scale(600);
              //draw object ft1
              ft1.draw();
              }
// if marker 2 is found
  if (ID == 2) {
              // turn things upside down to work intuitively for Processing users
              scale(1, -1);
              // hover the cube a little above the real-world marker image
              translate(0, 0, 0);
              //scale the size of the object to correct display size
              scale(20);
              //draw object ft2
              ft2.draw();
              }
// if marker 3 is found
  if (ID == 3) {
              // turn things upside down to work intuitively for Processing users
              scale(1, -1);
              // hover the cube a little above the real-world marker image
              translate(0, 0, 0);
              //scale the size of the object to correct display size
              scale(14);
              //draw object ft3
              ft3.draw();
              }
    }
}
```