

**Shenkar College of Engineering and Design**

**Faculty of Engineering**

**Department of Software Engineering**

**Q-BAR**

**Final project**

**By**

**Omer Goshen**

**Adam Galmor**

**Submitted as Part of the Requirements for receiving the**

**Bachelor of Science degree (B.Sc.)**

**27.11.13**

Gratitude

We would like to thank our project mentor and head of the Software Engineering department at Shenkar College of Engineering and Design, Dr. Amnon Dekel for his guidance in the past year, this project would never have been realized without his help.  
His and the departments’ faculty staff’s teaching and support were invaluable.

A special thanks to Tal Shustak of Intel Development Center in Haifa for who sought to use the college as a think-tank for finding emerging technologies use-cases.

And finally, the warmest blessing goes to our fellow student classmates and friends, for encouraging us along the way, good luck to you all!

Contents

[1. Introduction 1](#_Toc375322610)

[1.1. Overview 1](#_Toc375322611)

[1.1.1. Project Goal 1](#_Toc375322612)

[1.1.2. Problem 1](#_Toc375322613)

[1.1.3. Solution 1](#_Toc375322614)

[1.1.4. Audience 2](#_Toc375322615)

[1.1.5. Terminology 2](#_Toc375322616)

[2. Literature Survey 4](#_Toc375322617)

[2.1. Budweiser's "Buddy Cup" 4](#_Toc375322619)

[2.2. Guinness QR Cup 4](#_Toc375322620)

[2.3. Makr Shakr 4](#_Toc375322621)

[3. Architecture 5](#_Toc375322622)

[3.1. System Description 5](#_Toc375322623)

[3.2. System users 5](#_Toc375322624)

[3.3. Constraints 6](#_Toc375322625)

[4. Design 7](#_Toc375322626)

[4.1. reacTIVision 7](#_Toc375322627)

[4.2. Perceptual 8](#_Toc375322628)

[4.3. Web 11](#_Toc375322629)

[5. Implementation 13](#_Toc375322630)

[5.1. Overview 13](#_Toc375322631)

[5.2. Details 15](#_Toc375322632)

[5.2.1. PerC implementation 15](#_Toc375322633)

[5.2.2. Web-server Implementation 17](#_Toc375322634)

[5.2.3. Webapp Implementation 18](#_Toc375322635)

[6. Testing 20](#_Toc375322636)

[6.1. Overview 20](#_Toc375322637)

[6.2. Preliminary Study and Validation 20](#_Toc375322638)

[6.3. Verification 22](#_Toc375322639)

[7. Goals conclusions and further work 23](#_Toc375322640)

[7.1. Goals 23](#_Toc375322641)

[7.2. Conclusions 23](#_Toc375322642)

[7.3. Further work 23](#_Toc375322643)

[7.4. System Weaknesses 24](#_Toc375322644)

[8. References 25](#_Toc375322645)

[9. Executive summary and examples 26](#_Toc375322646)

List of Figures

[Figure 1 - Budweiser’s “Buddy Cup” 4](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322580)

[Figure 2 – Guinness QR Cup 4](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322581)

[Figure 3 – Makr Shakr 4](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322582)

[Figure 4 –6 examples of reacTIVision fiducials 7](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322583)

[Figure 5 - Kinect infrared coordinates map 8](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322584)

[Figure 6 - Depth perception using the infrared camera 8](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322585)

[Figure 7 – Marking a bar section 9](#_Toc375322586)

[Figure 8 – Homepage screenshot 11](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322587)

[Figure 9 – Social page screenshot 11](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322588)

[Figure 10 – Payment dialog screenshot 12](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322589)

[Figure 11 – Tab screenshot 12](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322590)

[Figure 12 – Menu screenshot 12](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322591)

[Figure 13 – Architecture overview (from the Final Project Presentation in Shenkar) 14](#_Toc375322592)

[Figure 14 – Flowchart for the bartender use-case 16](#_Toc375322593)

[Figure 15 - Webapp Class Diagram 18](#_Toc375322594)

[Figure 16 – Kinect Explorer view of a customer 21](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322595)

[Figure 17 – Kinect Explorer view of a barman 21](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322596)

[Figure 18 – reacTIVision’s camera and console windows in action 22](file:///C:\Users\ogoshen1\Downloads\FPD%20(2).docx#_Toc375322597)

# Introduction

## Overview

### Project Goal

The goal of this project is to provide a working implementation for a perceptual computing use-case; this prototype demonstrates how both a bartender and his customer can benefit from the use of technology.

It’s made up of several modules, allowing a customer to be identified and to place his order electronically, and it allows the bartender to serve to and bill his customer (at the same time), using gestures which are pretty much the same as his ordinary labor.

You might have seen robotic bartenders before, but this project is quite different, it aims to encourage human-human interaction, by making it more efficient and fun.

### Problem

Bartenders these days usually have to keep tabs on a computer. Those had touchscreens way before the majority of us had smartphones, and that makes sense. When you want things done in a bar or restaurant you can’t have the workers sitting around using the mouse and keyboard on ergonomic workstations, you want them with light feet, spraying warm smiles at your customers. Well today when most of us do have touchscreens in our pockets, they (the bartenders and waitresses) are still using the “same old thing”, they take our order, pour the drink, go and touch some prefabricated buttons on a touchscreen, and we get a little roll of paper with our tab on it.

And that’s the problem, by the time the bartender gets back, the customer can probably check his mail, update his Facebook status, twit some, and heck even play a round of Candy Crush… Bars can sometimes be places you want to go and drown your sorrows in, but that should be yours – the customer’s choice and not the by-product of slow inefficient service.

### Solution

The answer is to take the bar to the next level again, and go touch-less. That’s where Perceptual Computing technologies fit in, this project utilizes computer vision, facial recognition and hand gestures to help serve a drink faster than you can say Jack Daniels.

### Audience

The audiences of this project are mainly Hi-Tech fans looking for a smarter way of drinking. Try to imagine how bars in the near future would look like, most odds you are not imagining the same touchscreens being used today.

The speed of which technology is advancing nowadays is very fast. This demands us to think alike and see perceptual computing as a natural evolution of today’s user interfaces.  
We wish to use this NUI in order to provide the ingenuity needs of our fans.

### Terminology

* **Webapp** – the QBar mobile web application.
* **PerC –** Perceptual Computing. Usage of close-range hand gestures, finger articulation, speech recognition, face tracking, augmented reality experiences, and more, to make human interaction with computers more natural, intuitive and immersive.[[1]](#footnote-1)
* **Kinect** – is a motion sensing camera developed and manufactured by Microsoft for the Xbox 360 video game console and PCs. Based on a webcam-style add-on peripheral for the Xbox 360 console, it enables users to control and interact with it without the need of a keyboard, mouse, touchscreen or any outdate user control interface. This can be achieved through a Natural User Interface using gestures and spoken commands.[[2]](#footnote-2)
* **Gesture** - express an idea or emotion through bodily movements. It is a form of non-verbal communication in which visible bodily actions communicate particular messages, either in place of speech or together and in parallel with spoken words. Gestures include movement of the hands, face, or other parts of the body. [[3]](#footnote-3)
* **Q-Bar** – is a nickname for our smart perceptual bar.
* **Q-Bartender** – is a nickname for the bartender in the Q-Bar.
* **QR** – Quick Response code is a matrix code, 2-D bar code image, black and white Square that can be read by QR scanners and smartphones and mobiles phones with camera.
* **Fiducial** - In imaging technology, a fiducial marker or fiducial is an object used in the field of view of an imaging system which appears in the image produced, for use as a point of reference or a measure.[[4]](#footnote-4)

# Literature Survey



## Budweiser's "Buddy Cup"

This beer company has come up with a clever idea for another technological use for a beer glass.

Figure 1 - Budweiser’s “Buddy Cup”

These cups are equipped with a “bump sensor”, so when you tap your cup to another you and that person instantly become friends on Facebook.

It works by scanning a QR code on to bottom of the cup with a smartphone, which opens up Budweiser’s app, where you can link the cup to your profile.

## http://www.adweek.com/files/imagecache/node-blog/blogs/guinness-qr-code.jpgGuinness QR Cup

Figure 2 – Guinness QR Cup

The genius of this Guinness is that the QR code on the glass is literally activated by the beer.  
Pour a Carlsberg or Tuborg in there and you get nothing.

Scan the code with your smartphone, and it "tweets about your pint, updates your Facebook status, checks you in via Foursquare, downloads coupons and promotions, invites your friends to join you, and even launches exclusive Guinness content."

## http://images.gizmag.com/inline/makr-shakr-4.JPGMakr Shakr

Figure 3 – Makr Shakr

MIT’s Senseable City Lab is behind this robotic bartender, presented at Google’s I/O 2013 conference.

It’s actually much more than just that, users can mix their own drinks through an app on their smartphone and then share, learn and connect to others on social networks.

# Architecture

## System Description

This system aims to provide a Hi-Tech solution to the bartender bottle-neck pressure and the redundancy of the client’s social aspect as a result. You can call Ii the new “Hi-Tech flaring”.

## System users

There are only two types of users in the system, a bartender and his customers.  
They have a one-to-many relation dictated by the real-world, which to re-iterate the problem statement is what keeps the bartender “stuck” to his chores, tending the business layer instead of the social in his relations with the customers.

The user types along with a short description of how each can interact with the system are presented below, and will be elaborated as we refine the description of the project.

#### Bartender

This user is the only one tracked by the Kinect, his inputs to the system are his hands and coasters which he puts on the bar. He has a monitor and speakers to get feedback from the system.

#### Customer

Users of this type use the mobile web application on their smartphones for most of their actions, but can also wave their hand and show their face to the “hostess” (virtual, it’s a camera and face recognition module). While doing so, the application (a virtual hostess) replays with a QR representing of his account token from the server. The customer then scans the QR with his smartphone to boot up the web application, peered to his personal token.

## Constraints

The system has a few constraints; mostly they are technological and derive from the functional requirements of the project. Often these constraints actually help the developers converge to a solution instead of trying to pick from an almost infinite set of ways to solve the problem.

* **Budget** is a constraint, and although the project uses several hardware features, it is mostly a software solution, which is cheaper. Instead of fabricating beer mugs or assembling robots, computer vision, networking and algorithms do the heavy lifting.
* **Resources** are another important constraint; here they mostly amount to the hardware cameras and their SDKs. The Microsoft Kinect is the predominant in the long-range depth-sensing camera market, similarly is the   
  Creative Gesture Camera for short range. For the former there are two SDKs, Microsoft’s and OpenNI, for the later only the Intel PCSDK.  
  Since the MS Kinect SDK doesn’t have wrapper for Processing\Java using the SimpleOpenNI framework solves the constraint while the Intel PCSDK already has Java wrappers.

For recognizing coasters, Quick Response codes weren’t fit for job, as they require a larger size or closer camera positioning and better lighting conditions, hence the reactiVision fiducals.

* **Time** wasn’t a constraint for the bigger part of the project, but it was a factor while conceptualizing. The first steps of the project, recognizing and finding a solution for a Perceptual Computing use-case were held on a tight schedule, with weekly meetings of the students and academic mentors.  
  Planning is the most important stage, and can have deep effects on the architecture and design of the project.

# Design

The system contains several distributed modules, communicating over the network.   
The design is mostly the result of constraints (in hardware & software, as explained above, and elaborated specifically later on), but also follows best practices and methods of software development.

### reacTIVision

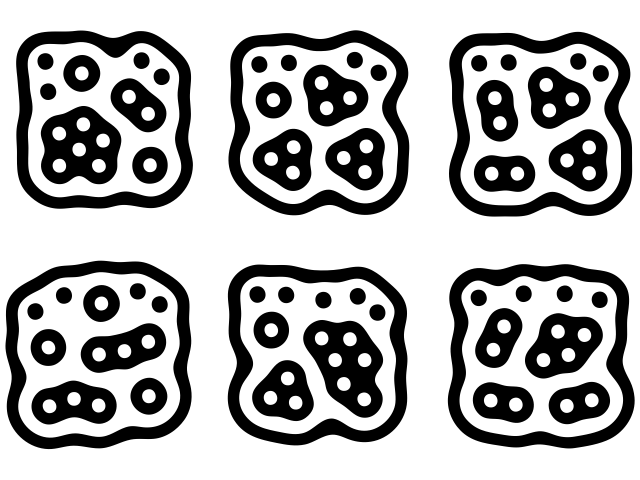
This is a proprietary piece of software that uses computer vision to identify special markers (*Fiducials,* Figure 4). It runs as a standalone application which sends [Open Sound Control](http://opensoundcontrol.org/introduction-osc) messages over pre-configured UDP ports. These messages contain the ID of the fiducial, its 2D position and orientation, and a timestamp.

Figure 4 – Six examples of reacTIVision fiducials

reacTIVision actually is the “brain” which operates the reacTable, an audio synthesizer with touch UI.

Fiducials were chosen over QR codes for their fast recognition and increased camera range.   
The poorly lit, something smoky environment of a bar makes computer-vision harder and also disqualifies solutions that use color. Fiducials are based on contrast and hence are color agnostic, which didn’t help implementation much (can’t really on color accuracy in pictures taken in bars) but provided nice eye-candy in the presentation with fluorescent yellow and magenta replacing white on the coasters.

### Perceptual

**Microsoft Kinect camera**

The Kinect contains three vital pieces that work together to detect your motion and create your physical image on the screen:

* **RGB color VGA video camera**the camera detects the red, green, and blue color components as well as body-type and facial features. It has a pixel resolution of 640x480 and a frame rate of 30 fps. This helps in facial recognition and body recognition.
* **Depth sensor**the depth sensor contains a monochrome CMOS (complimentary metal-oxide semiconductor) sensor and infrared projector that help create the 3D imagery throughout the room. It also measures the distance of each point of the player's body by transmitting invisible near-infrared light and measuring its "time of flight" after it reflects off the objects.

Figure 5 - Kinect infrared coordinates map

* **Multi-array microphone**  
  the microphone is actually an array of four microphones that can isolate the voices of the user from other background noises allowing users to use their voices as an added control feature.

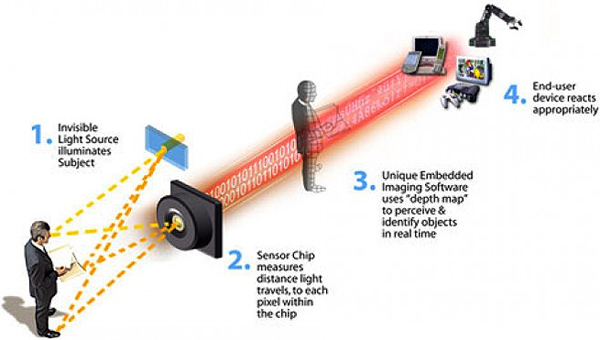
These components come together to detect and track 48 different points on each player's body and repeats 30 times every second.

Figure 6 - Depth perception using the infrared camera

**The OpenNI 3D sensing framework**

This module uses a Microsoft Kinect camera to track the bartender’s hands and recognize his gestures. It’s was written in Java\Processing using the OpenNI library.  
This module also serves as a client for the reacTIVision server, so only the combination of the bartender’s hand inside a bar section and a message from reacTIVision the coaster on the bar triggers the completion of a customer’s order.

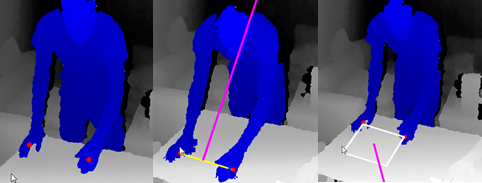


Figure 7 – Marking a bar section

The above images (Figure 5 – Marking a bar section) are screenshots of the running Processing sketch, they show a sequence of depth images from Kinect, with the tracked user colored in blue (courtesy of the OpenNI library). The user is performing a setup for a bar section, it’s consisted of marking the corners of a quad with his two hands.

Several gestures for performing this action were theoretically considered, neither were practical for implementation for various reasons.

Using machine-learning algorithms with a large training set is above the scope, expertise and budget for this project. There are several off-the-shelf applications, frameworks and tools which can help do this and “teach” the Kinect this two handed gesture. If they were ever to be used successfully in the future, they should probably trigger two events for the start and end of the gesture, containing 3D world-position coordinates of the hands making up the corners of the polygon.

Another possible method could be “dwelling” in the desired locations, while not nearly as hard to implement as the previously suggested method, it adds a lot of time to testing and usage.

For the results of marking a bar section to be accurate, meaning that is aligns to the plane of the bar, some extra work is needed. Once again, this is beyond the scope of this project, but a suitable method can be found in parts of various “whittling” solutions, those applications that reconstruct objects or environments when filmed by a depth-sensing camera such as the Kinect. It usually revolves around usage of the RANSAC algorithm to detect planes inside the point cloud.  
Once the bar plane is known, the section should be rotated to align properly.  
The sketch displays the section’s normal so the users can manually estimate the quality of the marking and repeat the process for better results.

As stated before, this project’s scope doesn’t span the features needed to make the gesture natural, intuitive nor immersive. The workaround was to use a mouse or keyboard, pressing the right mouse button (which when is a wireless mouse, a single user can hold in his hand while performing the gesture) or the spacebar on the keyboard (requires a second user).

There’s also support for deleting a section, for when the process results in a “bad” quad, which doesn’t align with the surface. The user can left-click a section with the mouse and press ‘delete’ to remove it.

### Web

Written in Java\Processing, this module uses the [Jetty](http://www.eclipse.org/jetty/) web-server and [Cometd](http://cometd.org/documentation/cometd-java) server-push library to host the mobile webapp. On the client-side, a responsive HTML5 webapp allows the customer to order drinks and pay his tab.

When the client reaches his seat he can decide his preferred way of placing an order. All he needs to do in order to use the webapp for it is to wave his hand from side to side. This will generate and display a unique QR on the screen in front of him, once the QR is scanned by a mobile device the webapp initializes with his unique token. You can see this process as a modern hostess in contrary to today’s busy human hostess.

#### User Interface

The webapp contains several tabs the first is the “Home” tab (Figure 6 – Homepage screenshot), its goal is to expose the user to the main social activities occurring at the local Q-Bar. It does that by displaying the currently playing song, the number of tweets, posts and check-ins made at the bar.   
For a worldwide point of view, the user can preview for example during a football game, the number of beers served at his favorite team homeland Q-Bar.

Figure 8 – Homepage screenshot

Figure 9 – Social page screenshot

Another social related tab is the last one named “Social”; its purpose is to display the users’ social networks events on a calendar, merged with highlighted Q-Bar events.

Below the calendar a QR is displayed, this is the same QR the user got upon his first interaction with the webapp. Its purpose is to enable a bill sharing system, if for example the client arrived with a friend to the Q-Bar and that friend decided he wants to pay the bill, he can then scan that QR and the bill tied to his friend’s token will be displayed on his mobile ready to be paid!

The second tab is the “Menu” (Figure 8 – Menu screenshot); it displays a list of beverages, their cost and an editable quantity input box. When the client presses the order button at the bottom of the page, the order is placed.   
The client is charged only when the Q-Bartender displays the right QR coaster to the image recognition camera.

This way the client can be pinged on delivery (Figure 9 – Tab screenshot).

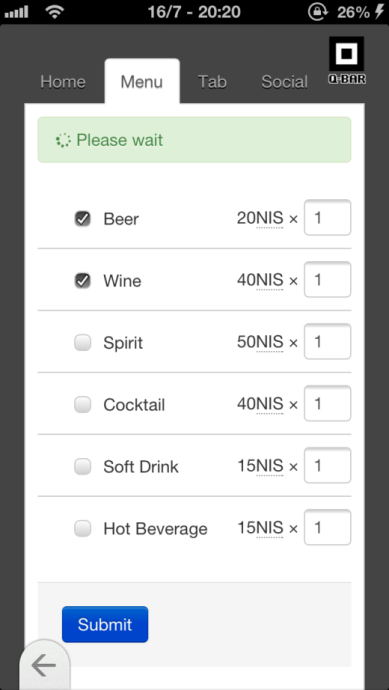
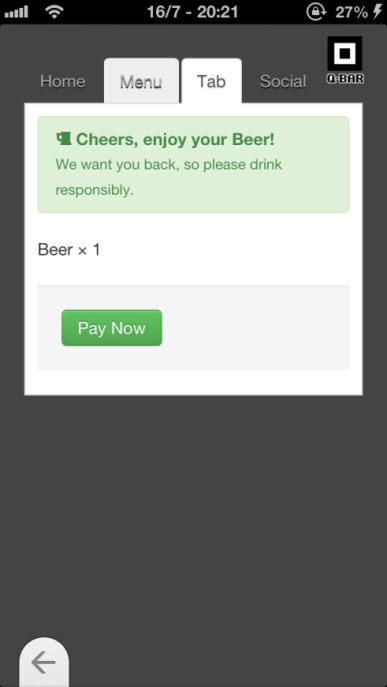
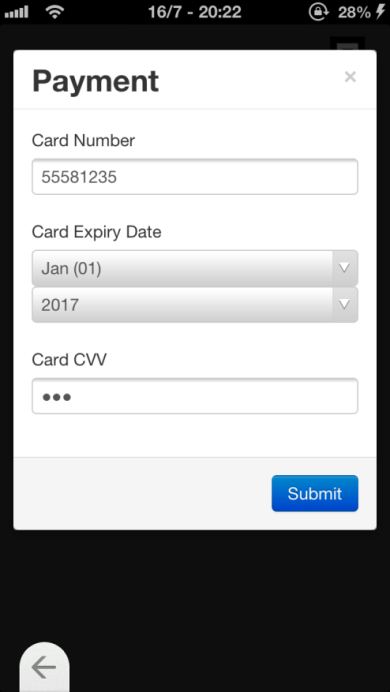


Figure 10 – Payment dialog screenshot

Figure 11 – Tab screenshot

Figure 12 – Menu screenshot

Upon charge the next tab “Tab” (checkout) gets focused describing the bill and enabling a payment option. The client can pay by entering his credit card credentials.

# Implementation

## Overview

An overview of the system is presented below (Figure 11); it shows the various components of the system with an illustration of their network topography.

Starting from the top of the diagram, on the left of the cloud you can see the various HTML pages that make up the QBar webapp, the arrows connecting them to the customers’ smartphones are bi-directional, which is an insinuation for the “Server-Push” technology used, it’s backed up the Java web-server (CometD) and uses a corresponding jQuery plugin.

Phones, representing unique users are actually distinguished through different codes; the QR codes are also bi-directionally connected to suggest the social aspect of the system.

On the bottom of the diagram, from left to right, we see the reacTIVision server connected to a regular webcam; it sends messages to a preconfigured host and port over UDP (which camera and ports are configured in camera.xml and reacTIVision.xml respectively, see the reacTIVision documentation for more information).

The perceptual computer is connected to depth-sensing cameras, which are continually queried for the users’ hands positions, gestures (such as waving a hand) and identity (in the form of facial recognition, whose usage wasn’t implemented in the scope of this project).

Finally, the web-server, hosts the various markup, stylesheet and script documents that make up the QBar webapp. It serves resources in a RESTful manner to AJAX requests made from the webapp clients over HTTP, again, using jQuery on the client.

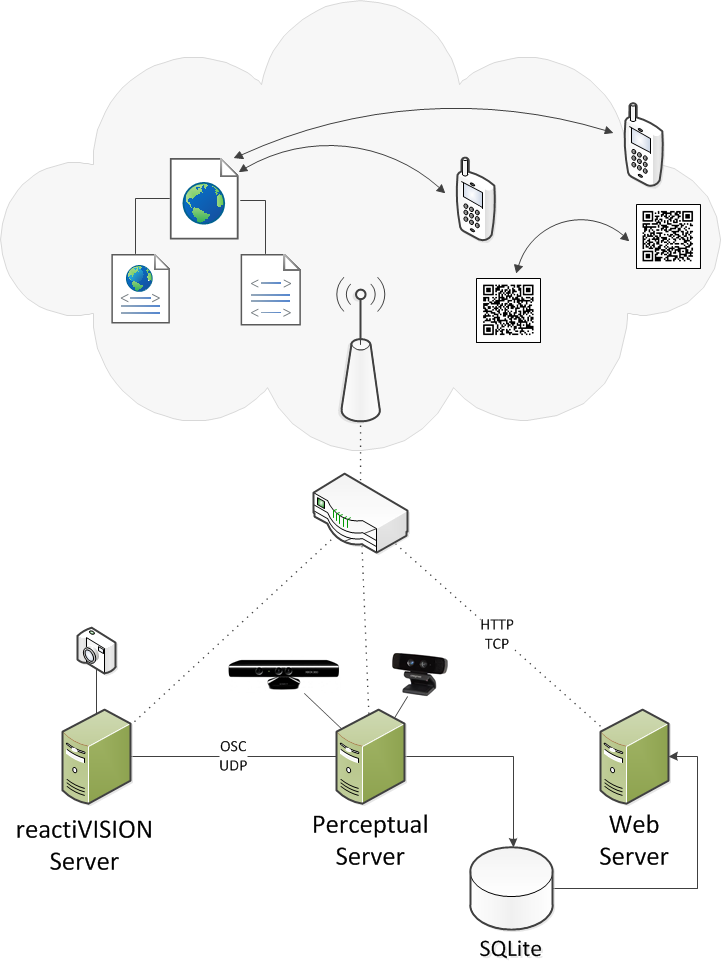


Figure 13 – Architecture overview (from the Final Project Presentation in Shenkar)

## Details

### PerC implementation

The main flow of system is illustrated below (Figure 12, Taken from the FRS document).   
It describes how the drinks served by the barman are tabbed.

Starting conditions are that the user is recognized by the Kinect and that bar sections are defined.

The system then goes into a loop, sampling the user’s hands and checking whether they are inside a bar section (elaborated later on).

When that check returns true, we can proceed to query the reacTIVision server for the presence of a coaster in the active section.

When both conditions are met, the drink will be tabbed to the current section.

As described in the Perceptual chapter of the Design section (4.2), bar sections are quads in the Kinect 3D world-space, to find the active one we calculate the Euclidean distance from the center of each section to the user’s hand. We pick the closest section, but only mark it as active when the distance is below a constant value.

#### Installation notes

To actually use the Kinect on a Windows PC, the OpenNI SDK has to be installed, along with the NITE middleware.

The SimpleOpenNI library wraps these for use with Processing (newer versions depend on the Microsoft Kinect SDK instead).

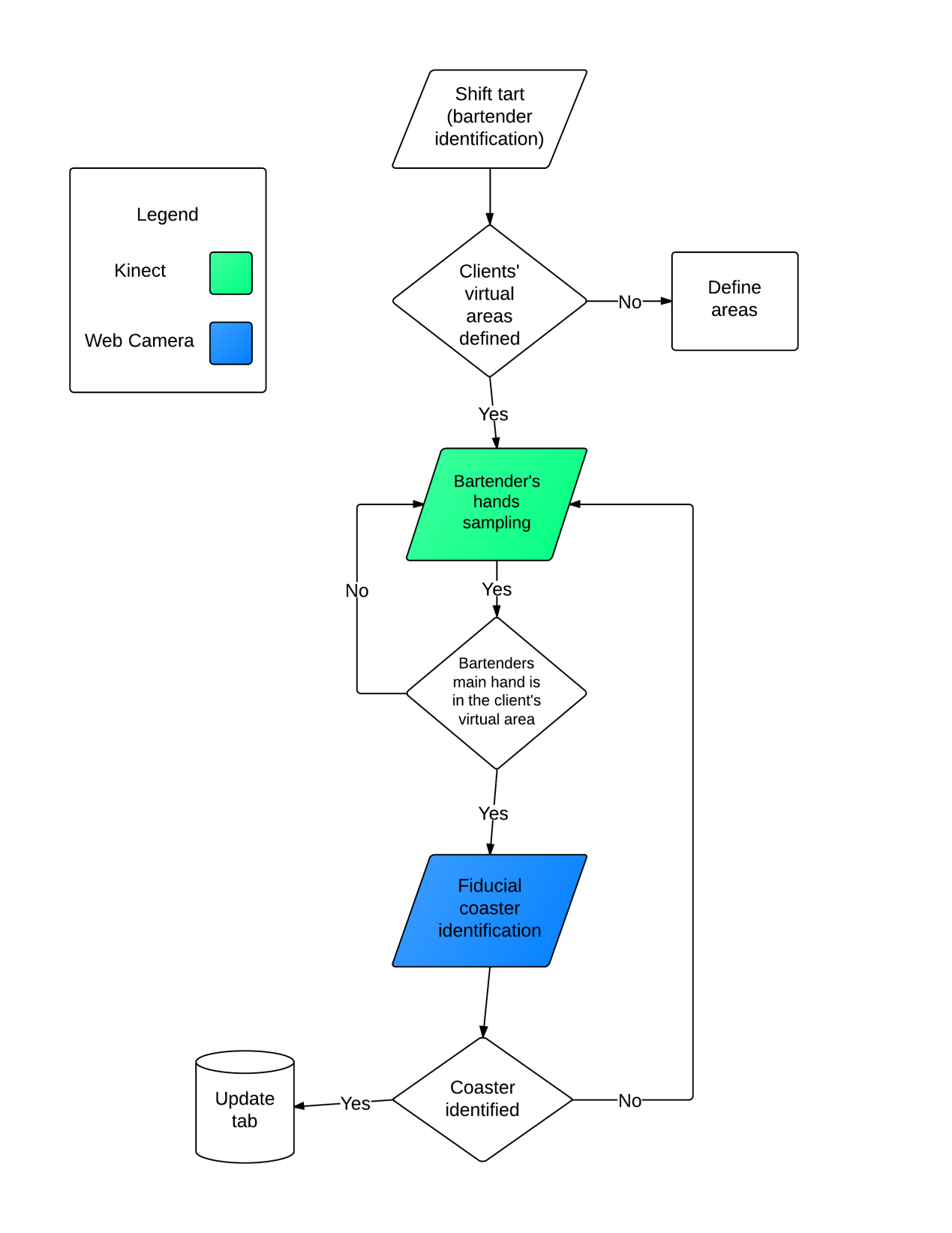


Figure 14 – Flowchart for the bartender use-case

### Web-server Implementation

The server for the web app was based around Jetty, which is a lightweight web server implementation in Java.

The various aspects of the server were coded in classes which handle the HTTP requests, the following is a description of the Handler interface implemented by these classes (From the Jetty API docs):

A [Handler](http://download.eclipse.org/jetty/stable-9/apidocs/org/eclipse/jetty/server/Handler.html) instance is required by a [Server](http://download.eclipse.org/jetty/stable-9/apidocs/org/eclipse/jetty/server/Server.html) to handle incoming HTTP requests. A Handler may:

* Completely generate the HTTP Response
* Examine/modify the request and call another Handler
* Pass the request to one or more other Handlers

Two handlers were written to deal with requests, the first responds to the ‘qr’ query parameter with a 250x250px PNG image of a QR code.  
The second returns a JSON representation of an order as sent by the webapp.

CometD is implemented as a servlet. Each request by the client spawns a thread on the server, which spins in a loop.  
The body of the loop checks whether the message from reacTIVision was received and if so, it echoes a JSON representing the drink and breaks.

The internals of the CometD servlet takes care of keeping the HTTP connection open, and only sends the headers of the response when all processing of the request is over (when the loop ends), this is how “server-push” is implemented.

### Webapp Implementation

A single-page-app HTML5 mobile application enables the customer to order drinks from a menu, get notified when the bartender serves it, and pay his bill.

The mobile web application uses Twitter’s Bootstrap framework for styling its presentation, and for several of its great javascript components (Tabs, Alerts, etc).

Backbone is a javascript MVC framework, and is used to organize the code and better separate presentational & business logic.

The model contains a single class representing a drink, and a collection of these to represent the customer’s tab.

There are several views, organized in a hierarchal structure. List views are made of individual items where each is bound to a model. In backbone you call the render function of a view, which based on a template (Underscore.js template, a utility library that’s shipped with Backbone), creates the proper HTML to display the model. In the above example, a <ul> tag for the list and <li>’s for the items.



Figure 15 - Webapp Class Diagram

A jQuery cometd plugin listens for server-pushed messages, those contain a collection of Backbone models in JSON format with the drinks that were served by the barman. These models end up in the customer’s tab page’s list view (Figure 11 – Tab screenshot).

There’s little usage of native javascript code in the webap, and for a good reason.   
Using industry-standard libraries of tested code is a must in these days of device and browser fragmentation.

Bootstrap’s CSS provides a unified user interface which along with jQuery’s selectors and events help you find your way around the DOM faster, manipulate it more sophisticatedly and free you from all kinds of browser quirks while doing so.

Not all features of the webapp are fully implemented, and some have views which show mockup data, especially those which are social-networking related. For more information about these unimplemented features the Further work section of the document.

# Testing

## Overview

Prior to any verification of the software was the need for feasibility tests and risk planning – validation to assure that the specs can be met, that such a product could be built, that it actually solves the problem and serves its purpose.

## Preliminary Study and Validation

As part of the requirements analysis and to even start using perceptual computing at a bar it was needed to check whether and how the equipment works in such scenarios.

Several assumptions were compiled to a list and tested, some of which at a real bar; the table below contains their description as well as the test results:

|  |  |  |
| --- | --- | --- |
| # | Test | Result |
| 1 | Kinect can recognize users even if occluded by a bar | OK |
| 2 | Kinect will work better in “Seated” mode | OK |
| 3 | Kinect voice commands will work despite background noise and music | FAIL |
| 4 | Kinect’s Infra-red imaging is unaffected by darkness \ colored lighting | OK |
| 5 | Kinect will not track reflected objects (like people in a mirror) | OK |
| 6 | The HD webcam has a high enough resolution to scan a coaster-sized **QR** code when placed above the bar | FAIL |
| 7 | The HD webcam has a high enough resolution to scan a coaster-sized **fiducial** when placed above the bar | OK |
| 8 | reacTIVision software is color agnostic and based on contrast only | OK |
| 9 | A single webcam’s field-of-view can cover the entire bar | FAIL |
| 10 | reacTIVision works with multiple webcams “out of the box” | FAIL |

From these test results the system’s architecture and design was derived, in what resembled a hybrid of the Waterfall and Spiral models, where each risk to be avoided was considered as a design failure and sent us back to the design stage. Mitigated and accepted risks did not delay the process as we were on a schedule to get working prototypes.

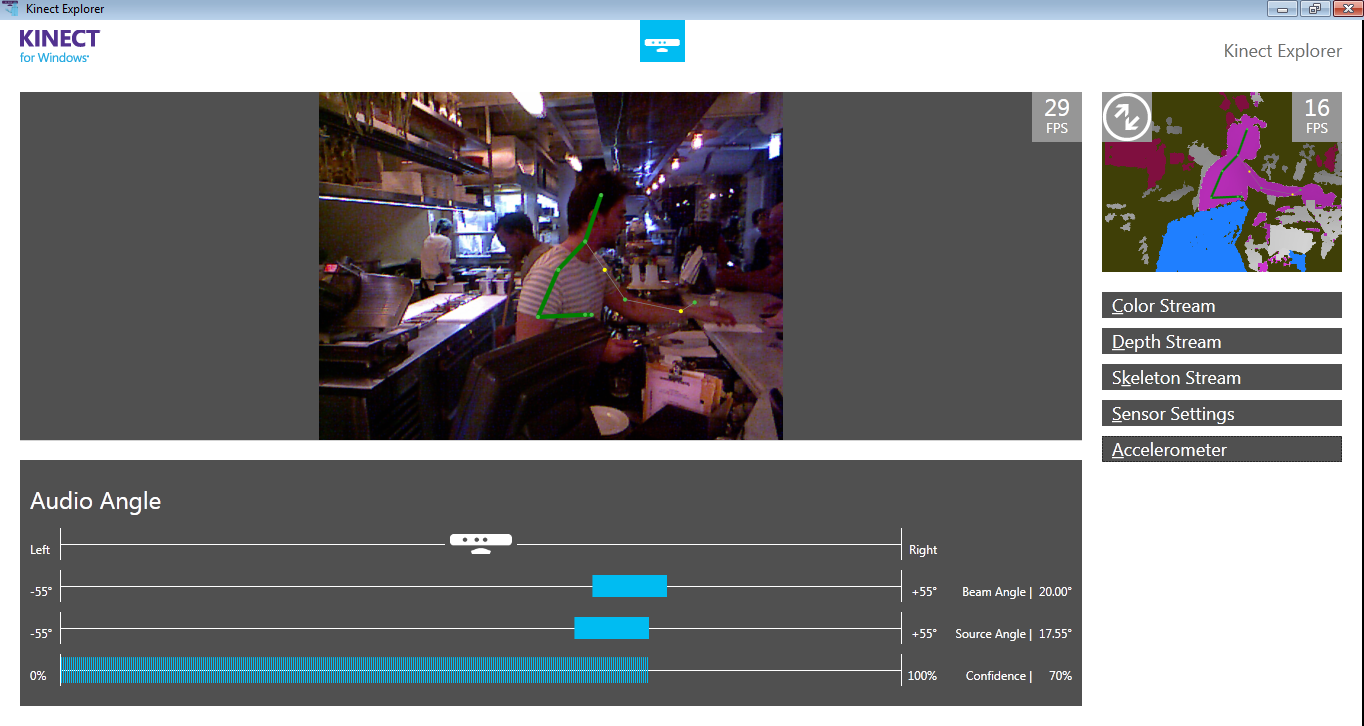
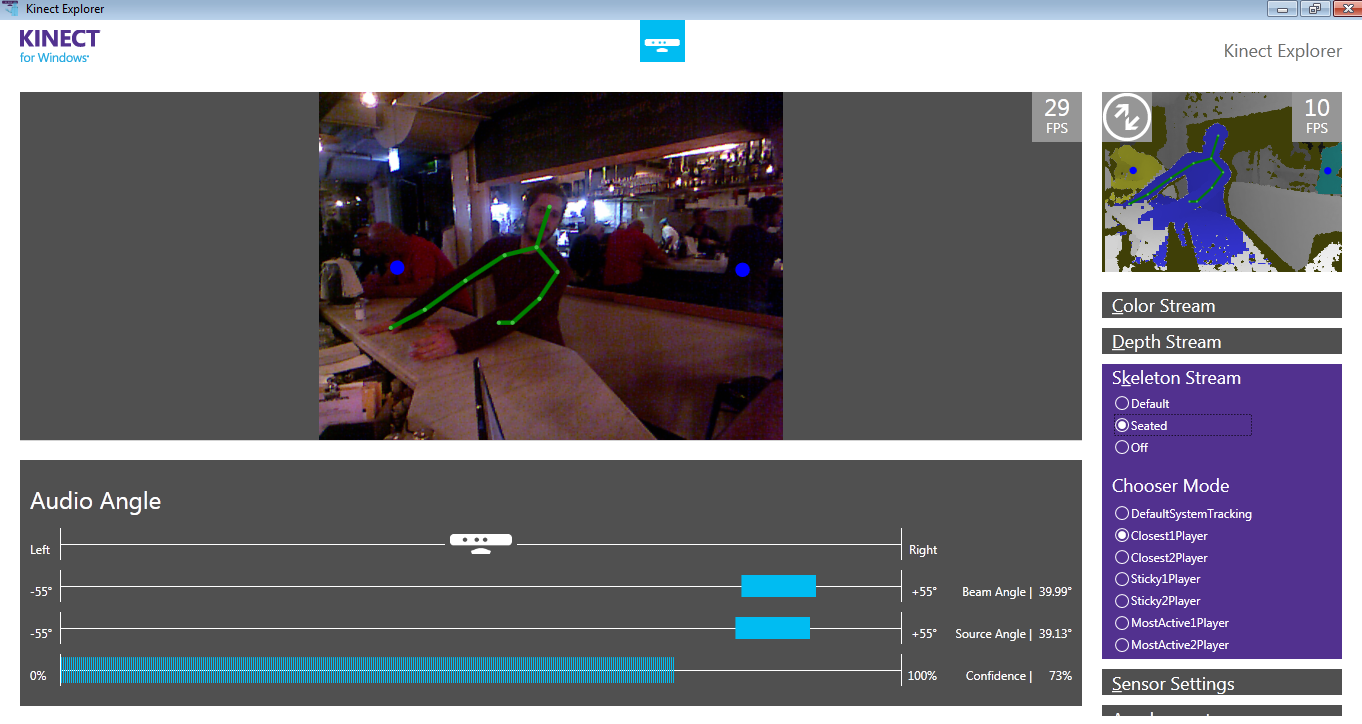


Figure 16 – Kinect Explorer view of a customer

Figure 17 – Kinect Explorer view of a barman

## Verification

To ascertain the quality of the software, in terms of correctness, completeness, efficiency and usability, it was necessary to test the individual modules as well as the system as whole.

Once the basic code infrastructure for the Kinect and reacTIVision was laid down, a “black-box” approach for testing was taken. Physical input – hands for Kinect and fiducials for reacTIVision was supplied, expected outputs were 3D coordinates for the first, and an integer code for the latter.

Next came a “white-box” testing stage, the inputs from the previous stage where fed into the Processing code where the expected output changed according to the system’s state (See Implementation).

Testing at this stage revealed for example, that the minimal comfortable distance to mark a bar section as active is when a bartender’s hand is less than 330mm close.

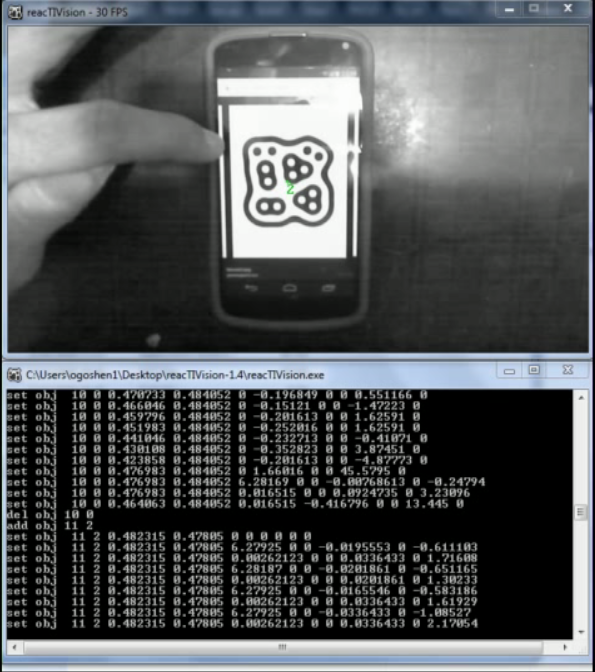
~~~~

Figure 18 – reacTIVision’s camera and console windows in action

# Goals conclusions and further work

## Goals

Increase the Q-Bartender’s client socializing aspect.

As a result of growing demand from the clients’ side for a faster and better service and on the other hand the Q-Bartender’s multitasking difficulties we’ve decided to relive him from nowadays’ touchscreens shackles and march him straight to the 21th century practical UI through the use of perception “gimmicks” and the social web application.

## Conclusions

We managed to achieve a more intuitive and immersive way of drinking and serving the client’s demands. The bartender no longer haves the need to go back and forth to touch screen placing his orders or generating the bill, to do so he can simply use perceptual gestures. As a result we belief the user experience of both the bartender and the client will increase.

In conclusion the bartender is free again to socialize with the customer making him feel interested. The client is aware of the surrounding social events and can choose wither or not he is willing to participate.

## Further work

* **Social rating of orders in the bar**To increase the client’s social immersion it is possible to add the ability to view what others have ordered, their drinking habits and drinks rating. In general the idea is to create for every customer a Q-Bar profile so that customers could see what others have ordered and how much they liked it.
* **All-you-can-drink feature with a special QR'ed glass bottom.**If we have an image recognition camera recording the customer nonstop, why not use it to watch the cup as-well. We can print the relevant drink’s QR at the bottom of the cup, once the cup is empty the QR is revealed and a ping is sent to the billing system, telling the Q-Bartender to refill the client’s cup.   
  This feature can be toggled by the web application or verbally through the Q-Bartender.
* **Network of Q-Bars**Have drinking contests with other Q-Bars worldwide, while watching a sporting event
* **Gamification feature**Grant users with experience point through the use of the system so they can latter on trade them for discounts and Q-Bar private events. This will encourage the regular customers to participate and break their technological phobias.   
  Receive experience points by using the web application to.
* **Image recognition camera theft security.**The stolen smartphone market is growing day by day, making people les free at the bar to socialize. It is possible to use a face recognition algorithm to attach the client’s smartphone to him. Once the user leaves his seat and the smartphone is no longer present inside the virtual customer area, an alarm will be raised a photo of the thief is captured so the Q-Bar security personal can trace and deal with the incident.
* **Voice recognition**The Kinect camera is installed with four microphones to reduce the surrounding noise and be able to perform voice command. This feature can be harnessed both for the client’s and Q-Bartender’s voice commands.

## System Weaknesses

* **The Kinect camera is limited to a seven meters vision.**Another vision disadvantage at the bar is the fuggy and dark environment, making it tough to recognize the drink QR coaster and perception gestures.
* Finding new and original hand gestures for the Kinect camera is not intuitive as it may seem.
* **The system as a whole is difficult to set up**Cameras of different nature should be placed in strategic points at the bar which can sometimes be missing or impossible to find.
* **Customers’ technophobic nature**Might repel the use of recording cameras pointing at them the whole time, capturing and documenting their every move. The social aspect can be intimidating, check-ins at the bar are public and can be seen by unwanted eyes.

# References

Figure 19 - Kinect infrared coordinates map

<http://graphics.stanford.edu/~mdfisher/Kinect.html>

Figure 20 - Depth perception using the infrared camera

<http://www.jameco.com/Jameco/workshop/howitworks/xboxkinect.html>

Figure 21 - Budweiser’s “Buddy Cup”:

<http://abcnews.go.com/blogs/technology/2013/04/budweisers-buddy-cup-bump-beers-become-buds-on-facebook/>

Figure 2 – Guinness QR Cup

<http://www.adweek.com/adfreak/guinness-qr-cup-reveals-scannable-code-when-full-140602>

Figure 23 – Makr Shakr

<http://www.makrshakr.com/>

Figure 24 –6 examples of reacTIVision fiducials

<http://reactivision.sourceforge.net/>

Open Sound Control

<http://opensoundcontrol.org/introduction-osc>

OpenNI

<http://www.openni.org/>

Processing

<http://processing.org/>

Jetty web-server

<http://www.eclipse.org/jetty/>

# Executive summary and examples

**The Concept:**This project started as a Co-op funded by Intel between the Industrial Design and the Software Engineering departments in Shenkar. We were looking for an innovative idea at the field of perceptual computing.

The first idea was a virtual closet, where the user can see on a screen an avatar image dressed with the cloths he picked.

After, we thought of a virtual desktop. We inspired the idea from movies like “Special Report” and “Iron Man”, where the lead actor uses holograms and gestures to control the computer. The idea was to create a virtual space and position elements in it. These elements can be viewed at the present with a tablet, but in the near future will be viewed with projecting glasses like the Google glasses.

At the end we thought of the Q-Bar and decided this idea can really put the touch screens at the past and bring the new spirit of perceptual computing into bars nowadays.

**The problem:**

Try to imagine how restaurant and bars around the world where before the touch screen revolution. Placing a new order demanded the bartender\waitress to remember, write down or at best to use a keyboard and a mouse. With touch screens today, placing a new order has become much simpler and faster than before saving lots of precious time.

We want to take this idea on to the next step by spearing the bar\restaurant employees the need to return to the cashier every time they want to place or update an order. With the use of gestures and perceptual computing, we can accomplish this goal. The ordering process is fluent and cool, providing a much more intimate and social feeling to the paying customers.

**Goals:**

The main goal we set for this project was to provide more social time between the customer and the bartender by eliminating the need from the bartender to go back and forth to the cashier and instead to provide a new “cool” way of placing an order.

With the use of image recognition algorithms and web cameras placed above the customer, the bartender flips the correct beverage coaster and reveals the fiducial QR to the web camera. To produce the bill all he needs to do is simply wave his right hand to the Kinect camera, point the relevant seat and the system will process and print the bill.

Another feature is the “Happy Hour” mode, the bartender at the start of his shift creates with two hands a virtual area for the virtual happy hour. Then when he places his hand inside it a theme can be heard and all prices get a discount.

**Related work and examples:**

Some would say this project is a-head of his time but we can already see a big growth in companies’ awareness of the perceptual field. During our final project presentation and at the beginning of this document we described a few:

* Budweiser recently launched the “Buddy Cup”, giving you the opportunity to make new Facebook friends by bumping your glass as you do a toast.
* Guinness published the “Guinness Cup”, a cup with a QR printed upon it. This QR is scannable only when the glass is full of Guinness beer, giving it the black background the QR needs.
* MIT University built the “Maker Shaker” bartender robot, where clients can upload\download cocktails recipes from and to him.

חשוב להדגיש כי שמנו לעצמנו מטרות אקדמאיות לההוכחת הרעיון (proof of concept), הצגת יכולות למידה, מקוריות ונזהרנו פעמים רבות שלא לגלוש לפן העסקי של הפרוייקט.

אנו מקווים כי הצלחנו להעביר את הרעיון והמסר של פרוייקט זה בצורה ברורה ומהנה, כזו הגורמת לקורא להרהר במחשבותיו לראות בדמיונו את חזונינו.



**שנקר – בי"ס גבוה להנדסה ולעיצוב**

**הפקולטה להנדסה**

**המחלקה להנדסת תוכנה**

**Q-Bar**

**פרויקט גמר**

**מאת**

**עומר גושן**

**אדם גלמור**

**מוגש כחלק מהדרישות לקבלת תואר ראשון**

**בוגר במדעים (B.Sc.).**

**27.11.13**

1. <http://en.wikipedia.org/wiki/Perceptual_computing> [↑](#footnote-ref-1)
2. <http://en.wikipedia.org/wiki/Kinect> [↑](#footnote-ref-2)
3. <http://en.wikipedia.org/wiki/Gesture> [↑](#footnote-ref-3)
4. <http://en.wikipedia.org/wiki/Fiducial_marker> [↑](#footnote-ref-4)