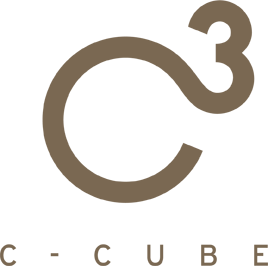
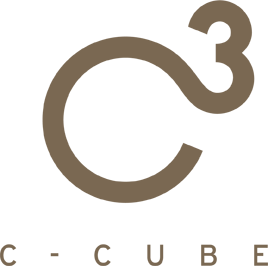
**Advanced User Interface 2015-2016**



*The Killer wears Prada  
Find out the Fashion Killer!*



**ABSTRACT**

The idea of our project is the implementation a game based on a simple treasure hunt located in a fashion Shop. The goal of this game is to find out the outfit of a killer thanks to the clues given by some items present in the game which corresponds to real clothes in the shop. The interaction is between consumer and game is managed using a Kinect.

With our project we would like to address the needs of the shop owner (attract more consumers, increase sales, new items discovery) in a funny and simple way that also allows consumer to be entertained while shopping.

Final Presentation

Presented By **C-cube TEAM**

Caielli Andrea Luigi Edoardo [andrea.caielli@mail.polimi.it](mailto:andrea.caielli@mail.polimi.it)

Cecere Monica [monica.cecere@mail.polimi.it](mailto:monica.cecere@mail.polimi.it)

Ceruti Federico Maria [federicomaria.ceruti@mail.polimi.it](mailto:federicomaria.ceruti@mail.polimi.it)

Table of Contents

[**Introduction** 4](#_Toc442798431)

[**User Requirements** 5](#_Toc442798432)

[**Stakeholder vs Need/Goal table** 5](#_Toc442798433)

[**Stakeholders** 5](#_Toc442798434)

[**Needs & Goals** 5](#_Toc442798435)

[**Requirements** 6](#_Toc442798436)

[**Scenarios** 7](#_Toc442798437)

[**Scenario 1: First Time** 7](#_Toc442798438)

[**Scenario 2: Parallel game** 10](#_Toc442798439)

[**Scenario 3: Winning** 13](#_Toc442798440)

[**Scenario 4: Registration** 14](#_Toc442798441)

[**Demo Scenario** 15](#_Toc442798442)

[**Differences between prototype and real solution** 15](#_Toc442798443)

[**Game Logics** 16](#_Toc442798444)

[**Game concepts** 16](#_Toc442798445)

[**Game rules and mechanics** 17](#_Toc442798446)

[**Population** 17](#_Toc442798447)

[**Implementation** 19](#_Toc442798448)

[**Database** 19](#_Toc442798449)

[**Database interface** 20](#_Toc442798450)

[**Classes** 20](#_Toc442798451)

[**Helpers** 22](#_Toc442798452)

[**Controllers** 25](#_Toc442798453)

[**Interface** 29](#_Toc442798454)

[**Feasibility study** 31](#_Toc442798455)

[**Technical Analysis** 31](#_Toc442798456)

[**Hardware** 31](#_Toc442798457)

[**Software** 31](#_Toc442798458)

[**Database** 31](#_Toc442798459)

[**Timetable** 32](#_Toc442798460)

[**Business Analysis** 33](#_Toc442798461)

[**Profits** 33](#_Toc442798462)

[**Costs** 33](#_Toc442798463)

[**Risk Analysis** 35](#_Toc442798464)

[**Conclusion** 36](#_Toc442798465)

[**Project decisions** 36](#_Toc442798466)

[**Main issues** 36](#_Toc442798467)

[**Workstation constraint** 36](#_Toc442798468)

[**Further Works** 36](#_Toc442798469)

[**Why Choose US** 37](#_Toc442798470)

# **Introduction**

The idea of our project is the implementation a “find the clue” game based in a fashion Shop. The goal of this game is to find out how a killer was dressed thanks to the clues given by all the items present in each game’s room. Each item correspond to a real cloth in the shop. By collecting real clothes in the Shop you get them in the game inventory and then you can combine them, accordingly to the clues, create and submit the identikit.

The interaction is between consumer and game is provided via Kinect.

With our project we would like to address the needs of the shop owner (attract more consumers, increase sales, new items discovery) in a funny and simple way that also allows consumers and their partners to be entertained while shopping.

This document first identifies the Stakeholders and their needs/goals, describing them in relation to what we have to guarantee in the implementation.

Then we propose some explicative scenarios that have paired images (conceptual screen vs real game screen) to show the basic mechanics of the game.

After that we enter in details with the game logic analysing them, with particular reference to the Randomization phase.

Next comes the Implementation part, which describes in details how we implemented the project. Here we discuss about Classes, Controllers, Helpers and DB and how they interact between each other.

Following there’s a feasibility study underlying the technical requirements, the planned timetable and the Business analysis of predicted costs and profits. We also added a Risk Analysis.

Finally we propose and argument the advantages of choosing our project.

# **User Requirements**

## **Stakeholder vs Need/Goal table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Goal/Needs  Stakeholders | Entertainment | Advertise | Increase selling | Attract clients | Item Discovery |
| Owner | 1 | 3 | 3 | 3 | 3 |
| Employee | 0 | 1 | 2 | 0 | 1 |
| Customer | 3 | 0 | 0 | 0 | 2 |

3 – High interest rate

2 – Medium interest rate

1 – Low interest rate

0 – Not applicable

### **Stakeholders**

**Owner:** the stakeholder that owns the shop or the manager of the shop.

**Employee**: all the employees of the shop, from the ones more directly involved (registration, etc.) to the ones that have to put the items away.

**Customer**: all the shop’s customer, from the active shoppers to those who accompanies them.

### **Needs & Goals**

**Entertainment**: the costumer has to be amused by the game.

**Advertise**: we mean both the advertisement of the shop and of its products.

**Increase Selling**: increase the number of products sold and so increase profits.

**Attract Clients**: increase the number of costumers.

**Item discovery**: tell the user which clothes are in the shop and spur the user to explore the shop.

The game has to be entertaining in order to satisfy the consumer. If the game is enough funny, the consumer will return more frequently or advice its friends to come, augmenting the number of consumers. By proposing new items to the consumers and inciting them to explore, the shop’s selling can increase as well as profit.

## **Requirements**

* The game has to be fun and not so difficult in order to entertain the customer, but enough challenging
* The game has to show as many items as possible
* The game has to spur the consumer to cover as many shop’s sections as possible
* The player should be able to access the game at every time while playing
* The player should be able to share its experience

All the interactions and architectures are displayed in the following scenarios in order to allow a better comprehension of the system.

# **Scenarios**

## **Scenario 1: First Time**

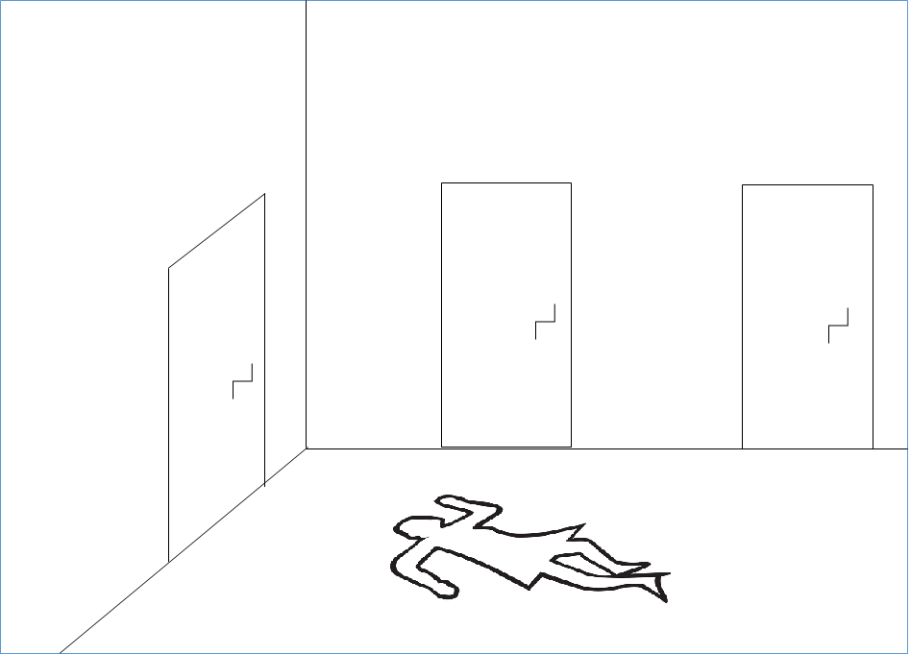
Bob enters the shop accompanying his wife Alice. To avoid boredom he goes to the game workstation.

After registering himself, (scenario 4) he enters. The Kinect recognizes his QR-code and the starting screen appears:

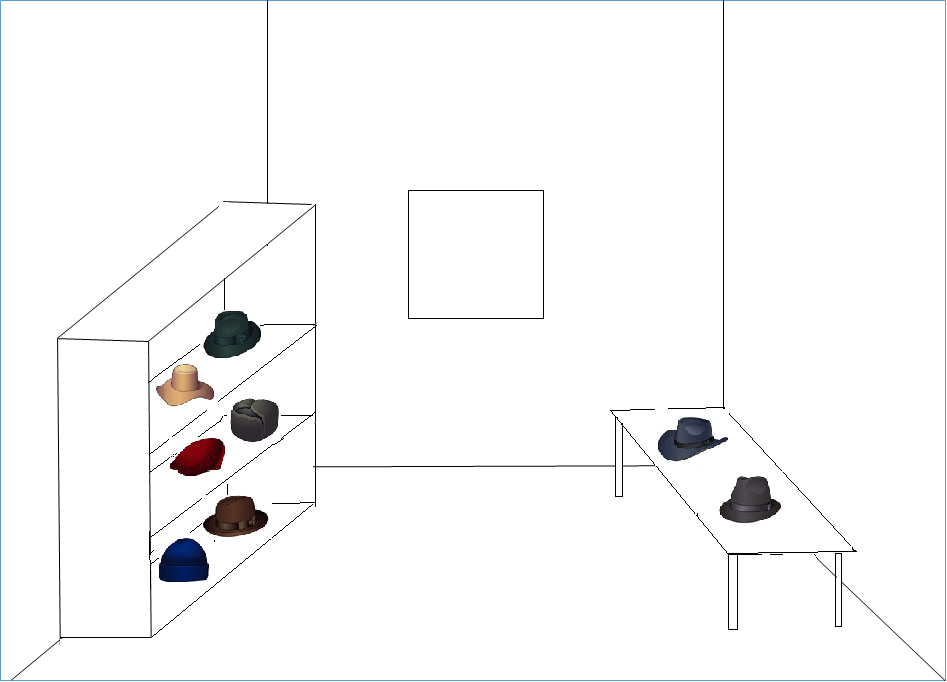


He decides to start and using the Kinect presses the Start button.

It goes to the first game screen, showing a hallway with some closed doors:



By making the right move, he opens the first door and the room is shown:



Navigating through the room using the Kinect, he discovers the first evidence. By pressing on it its photo and description appears, plus a clue:

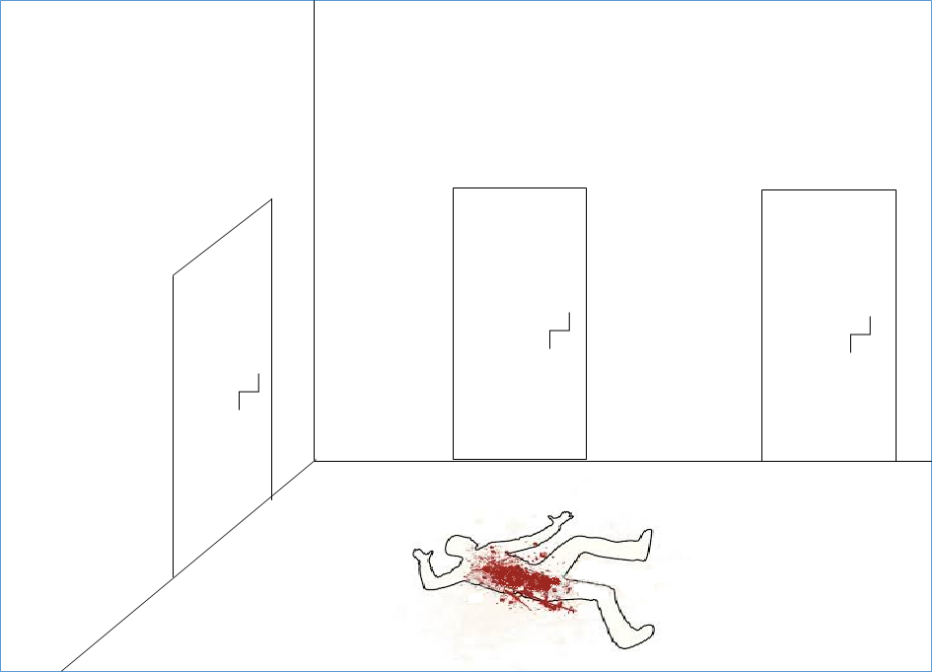


He decides to go looking for that object. The smart lights of that section will switch when he leaves the game stand.

## **Scenario 2: Parallel game**

Carla and her friend enter the shop looking for a dress. Unable to find it, they decide to try the new game offered by the shop.

After registering themselves, (scenario 4) they enter. The Kinect recognizes her QR-code and the starting screen appears:



Then after the first screen, they enter the first room. They collect a couple of cloths and explore a couple of rooms.

Navigating through the last room using the Kinect, Carla discovers another evidence. By pressing on its a photo and description of the products appears, plus a clue:

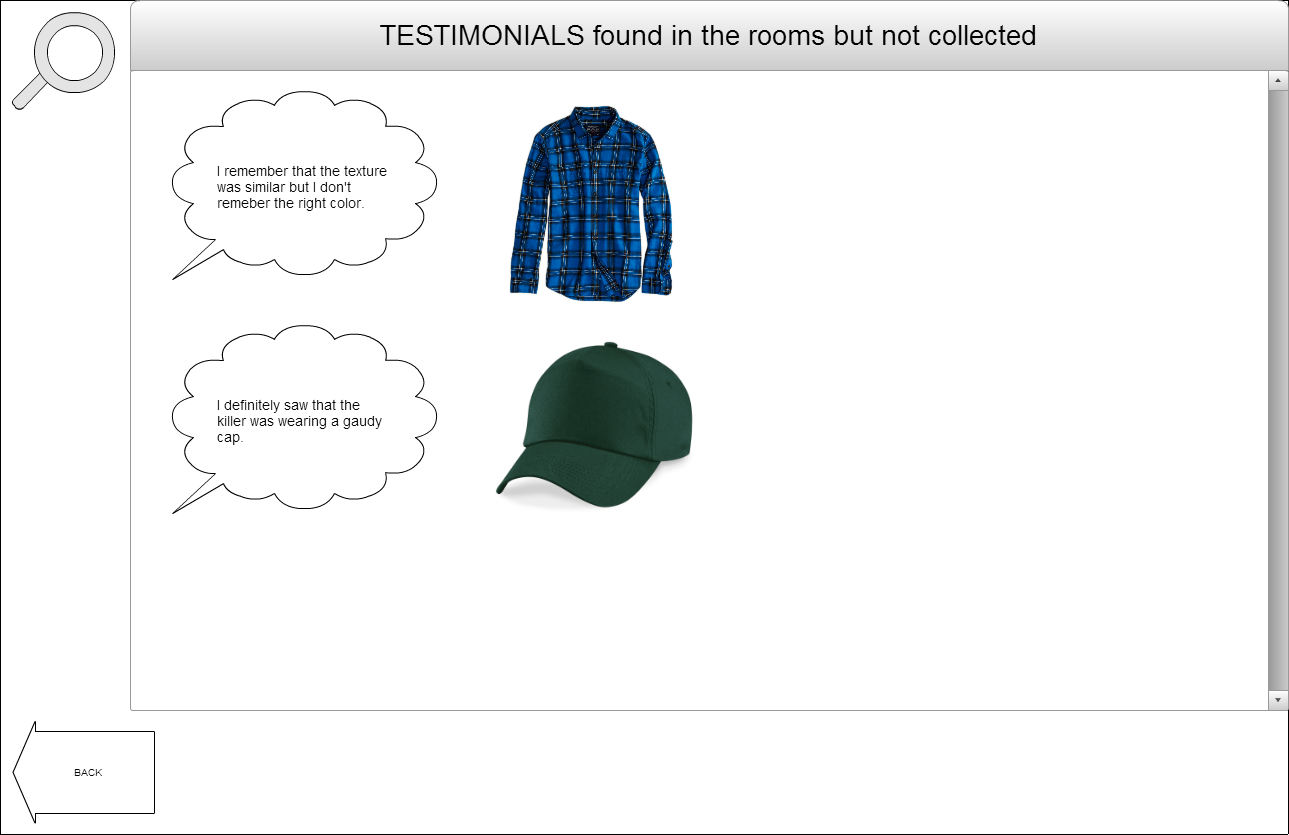


She decides to go looking for that object.

Carla returns since she forget what she was looking for because, during the search, she had found out a lovely skirt and wants to buy it. She returns to the Kinect, she lets the system recognize her QR-Code and then the room that she visited last appears. She accesses her inventory:



With a Kinect gesture she presses the button to see the testimonials and to remember the object she was looking for previously:



Now she can go looking for the object again.

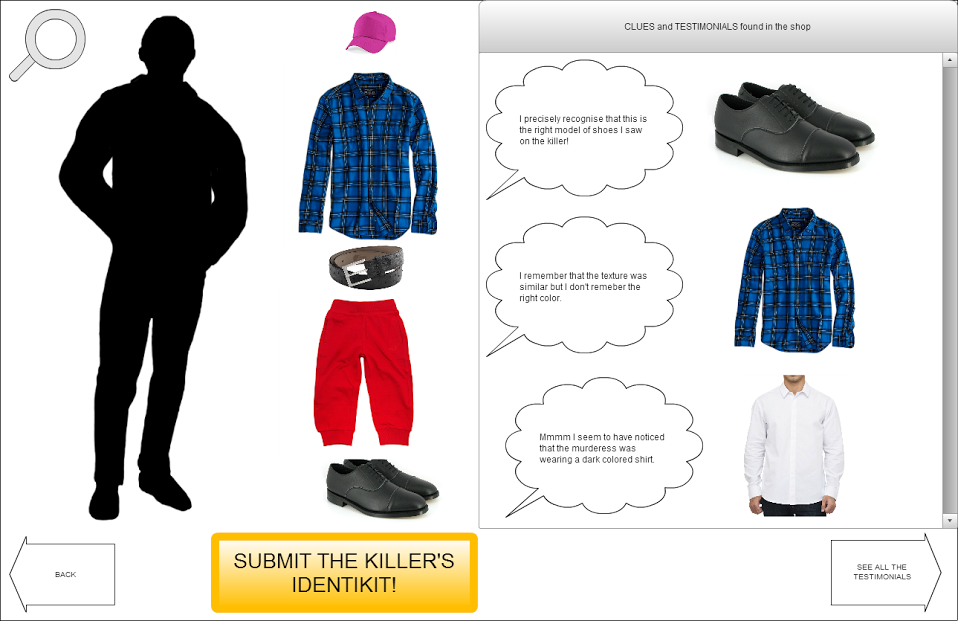
## **Scenario 3: Winning**

After finding the last possible clue, Bob returns to the Kinect.

He lets the system recognize his QR-Code and then the room that he visited last appears.

He checks if the cloth is right, and then adds it to its game inventory.

He is now able to propose a solution to the game, so he goes to the “identikit screen” and creates the killer outfit by dragging the right cloths from the clues and testimonials portion on the black silhouette:



He then submits the solution and… it is correct! He gets the Victory screen.

Then its discount is printed, he can take a photo with that outfit and share it on Facebook:



His wife has ended her purchasing at last. He now can use his discount.

## **Scenario 4: Registration**

Diana, an employee, inputs in the game system the players data (name, surname…) and prints out the QR-code related to the user. The QR-code allows the user to play the game and it is used to identify him/herself.

## **Demo Scenario**

Here we put the full scenario we used for demo

Andrea goes int the cloathes hop and decides to try their new game. He enrolled in a specific desk and the employee gave him a QRcode.

Andrea goes to the Kinect workspace and waits until the previous player leaves. Then Andrea steps forward and places itself on the sign in front of the Kinect. He waits until its QR is recognized and the starting screen appears.

------------

After reading the rules Andrea press the Start button and the game commences. A Hall with 3 green-glowing doors is shown.

(magari ci diciamo anche come viene gestita la creazione delle stanze a runtime)

Andrea decides to visit the room behind the first door on the left. After pressing the door a []room filled with [] is shown.

Andrea selects the first Item, the …… . After pressing on it and a new screen appears.

This screen shows all the details about the real cloth, a photo and a clue which, combined with the others 2 that Andrea will discover during the game, allows to identify the correct Item for that room (or that kind, which is exactly the same since one room contains one and only one kind of item).

Andrea repeats the same procedure for all the items in the room, and the following 5 screen appears:

As already pointed out Andrea is able to collect only 2 clues for this item kind.

Andrea now decides to look at its inventory to see a recap of all the properties that a correct cloth has to satisfy. By pressing on the Inventory button the inventory screen appears

Now that Andrea knows which items he needs to look for, the player leaves the workstation and starts the searching.

After a while Andrea has collected 4 items ::::::::

The player takes them in front of the Kinect and shows their barcode one at a time:

1 ------------------- giusto

Andrea returns to the hall and notices that now the light around the newly visited room is yellow

Andrea shows other barcodes

2--------- non c’entra

3-------sbagliato

4------------2° giusto

After the recognition of the 2nd right item, the 3rd resolutive clue for the room is shown

Andrea leaves the room and returns to the Hall. The player notices that the door closing the room he just finished glows white.

The player then repeats the aforesaid procedures for every single room until he disclose all the resolutive clues.

Andrea now goes in the Inventory to create the Identikit of the killer.

The player notices that the item … doesn’t fit any clue and decided to discard it. He selected it and then presse the…button.

## **Differences between prototype and real solution**

In order to allow a better game playability to we relaxed some of the original rules.

Now each room always contains the same kind of items, which allows many simplifications when populating the rooms.

We added a Trash bin, which is reachable from the Inventory, where is possible to put items the player. It’s possible to empty the trash (removing all elements) or restore an element. Once an item is removed, to add it again to the Inventory the user had to take its barcode in front of the Kinect again.

Clues are assigned randomly to each item, not according previous discovered clues. This because we want the user to look at each Item, since one of the needs to address is Item discovery.

The player no longer needs to be in the right room to add Items to his inventory, it is enough that the items he/she brings in front of the Kinect is one of the depicted ones in order to add it.

# **Game Logics**

This section covers all the aspects related to game logic. Here we explain the Game concepts and rules; then the population procedures and finally we clarify some other aspect of the game.

## **Game concepts**

1. **Item**

The item is the in-game representation of a cloth present in the shop

1. **Item Kind**

The item kind represents the kind of cloth that the item represents (ex. Trousers, Shirt etc.). For our demo we used 3 Item kinds: Hat, Trousers and T-shirts.

1. **Clue**

The clue is a characteristic/property of the item. We defined 4 properties: Color, Gradiation, Shape and Texture.

* 1. Color

For the Demo we insert 8 colors: black, blue, brown, green, red, purple, yellow and white.

* 1. Gradiation

The gradiation represents the color gradiation of the item. It can be ‘light’ or ‘dark’.

* 1. Shape

The shape tells us if an item is long or short. Its meaning varies based on the kind of the item. As an example, if the item is a shirt it means long or short sleeves. A particular case regards ‘hat’ item kind. For those items ‘long’ means they are wide-brimmed hats (like Fedora) while ‘short’ means they are Caps.

* 1. Texture

The texture represents the texture of the cloth. We added 5 textures: flowers, pois, stripes, scottish and plain-color.

1. **Room**

The room contains Items of the same kind. For the Demo we created 3 rooms.

1. **Inventory**

The inventory contains a list of all the clues already discovered and a collection of recognized Items (items whose Barcode had been recognized by the Kinect). From here it’s possible to discard unwanted items in the trash and to create an Identikit.

1. **Identikit**

The identikit is the reconstruction of the outfit of the killer.

## **Game rules and mechanics**

This game has fairly simple rules, which are the basic rules for a treasure hunt. Each item reveals a clue about the correct item, which is one of the items shown in the room. After collecting enough clues to identify the 2 candidates “right” items, the player has to collect the real clothes associated to them. He will be able to do so because each item, when selected, shows a photo of the real cloth plus some information about it. Once the cloths are collected, the player has to show their Barcodes to the Kinect, which will recognize them and add them to the Inventory. When all the 2 candidate “right” items had been added, a resolutive clue is shown. Whit this clue the player will be able to correctly recognize the right Item when she will have to create the Identikit. This process is repeated for every room. The Inventory is accessible at any time. From the Inventory items can be removed by putting them in the trash. When all the items and clues had been collected, the player has to access his inventory

## **Population**

All the possible combinations of Kind and properties made up more than 400 possible combinations. We decided that each room contains always the same kind of item, which reduces the number of combination to 140. This is still a big number so for this Demo we adopted some tricks to reduce the amount of possible combination and lower them to 75 per room.

In order to extract correct game items a semi-randomic population algorithm had been used. This algorithm is based on the fact that we start by randomly extracting the properties that the right item has to have.

So first we create the clues by randomly choosing 3 out of 4 possible clues. Two of this clues (called 1 and 2) are given directly for the items in the room and allows the user to identify the 2 “most correct” items, while the third one called 3) is the deciding one, which allows the user to choose the correct item. Then we extract from the DB the items, following this table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kind of Item | Clue 1 | Clue 2 | Clue 3 | Clue 4 |
| A | true | true | true | don’t care |
| B | true | true | false | don’t care |
| C | true | false | true | don’t care |
| D | true | false | false | don’t care |
| E | false | true | true | don’t care |
| F | false | false | false | don’t care |

A is the correct solution, which means that this item corresponds to all the clues we extracted. B is the second “more correct” item.

Once items had been chosen, each of them gets a sentence that states the clue (like “the correct item is RED et.) in a random way.

This procedure guarantees that the population of Items in each room changes for every new game while following the game rules.

# **Implementation**

This sections covers all the implementation aspects of the project. Since no specific Hardware is needed, this section is only about software implementation.

The project was implemented using Microsoft Visual Studio 2015, and so we used C++ language.

For this project a MVVM (Model-View-View Model) pattern had been used. This means that the project is highly modular, easily expandable and adaptable.

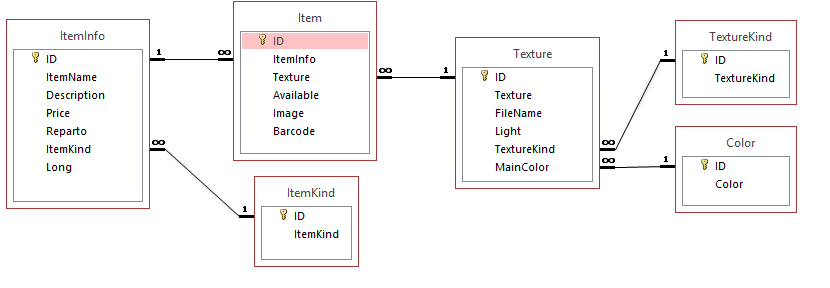
This chapter only covers View Model and Model, while View is detailed in the following chapter (Interface).

To better clarify components structure and their interactions, UML schemas had been provided.

## **Database**

Database are a critical aspect of modern applications, and this one is no exception. The main difficulty in was to create a DB structure that was able to interface with an already-existing consumer DB without having to change the whole structure of the project.

In order to accomplish this we adopted the following DB schema:



ItemInfo represent the hypothetical DB provided by the shop owner, which contains general information on the real cloth.

ItemKind contains the kind of outfit of an item (ex. Trousers).

Item is the core table, the one that works as a bridge between the graphical representation and the real data of the items. Here the item’s info are connected to their textures (graphical representations).

Texture contains the information on the graphical aspects of the item. Combined with Color and TextureKind it states the main color, the texture and the color gradiation of the item.

Our “demo” DB contains 302 items whose content are real word ones (each of their photo is a real one that had been taken from the Internet).

This separation between graphical and general info allows us to manage the consumer information (stored in ItemInfo) apart from the graphical information we used in for the in-game representation of the items. This also means that with an easy (even if not short) work we should be able to adopt this structure for every kind of DB our consumer provides us. Notice that a DB adaptation may not be necessary thanks to this interface.

## **Database interface**

The Database interface allows the communication between the real DB and the program. This allows the program to see the DB as a BlackBox, which allows us to ignore the DB characteristics (language, structure) with respect to the program. Moreover this contributes in the creation of a modular structure of the project, which is unbounded from the DB structure that is used. It will only suffice to change the interface in order to adopt the program to different. In addition to that, we can also address the issue of safe DB queries (better discussed in Risk Analysis paragraph) since we do not allow direct connection between DB and program.

## **Classes**

Classes creates the model structure of the game. Classes represents the objects of the game. The main classes of our projects are:

1. **Game**

It is a container class that represent the actual state of the game. This class contains all the objects from other classes is serializable and it’s used to save the state of the game when the player switches.

1. **AbstracItem**

Contains the graphical properties that an object had to have in order to be retrieved from the DB, which means Color, Gradiation, Shape, Texture and ItemKind. It is used in the population phase.

1. **Item**

This class is the software representation of the item in the game. It contains all the information about the item retrieved from the DB (its description, price, and data). It extends an AbstracItem, so contains also the graphical properties. It also contains the Clue property

1. **Item Graphical Properties**

For each of the aforesaid graphical properties, it contains the effective value of the property.

1. **Player**

The player class represents the player in the game. It’s related to the player table in the DB. It contains the player’s Username, QRCode and score.

1. **Room**

This class is the implementation of the game Room. It contains all the items and the last, and resolutive, clue.

1. **Solution**

This class contains object of AbstracItem kind that represents the graphical properties of the correct Items.

**Implementation of other Game Concepts**

1. **Trash**

The concept of Trash is implemented with a simple Boolean variable (IsTrashed) assigned to an Item object. Each time the Trash window is opened, a loop loads all the Items with the IsTrashed attribute set at true.

1. **Inventory**

The concept of Inventory is implemented with a simple Boolean variable (IsInInventory) assigned to an Item object. Each time the Inventory window is opened, a loop loads all the Items with the IsInInventory attribute set at true.

1. **Clue**

The clue concept exists as property contained in the Item object. As explained in the rules, every time a player clicks on an Item, the rule related to the item is retrieved from the related Item object by asking the clue property value.

The clue is translate as a string and it’s corrected with respect to the ItemKind.

## **Helpers**

Helpers are classes that creates an interface between game and external API. Helpers works like libraries that provides methods to use. In our project we have the following helpers:

1. **DB Helper (Database interface)**

This helper manages all the queries that had to be executed on the DB. The

1. **QRreader Helper**

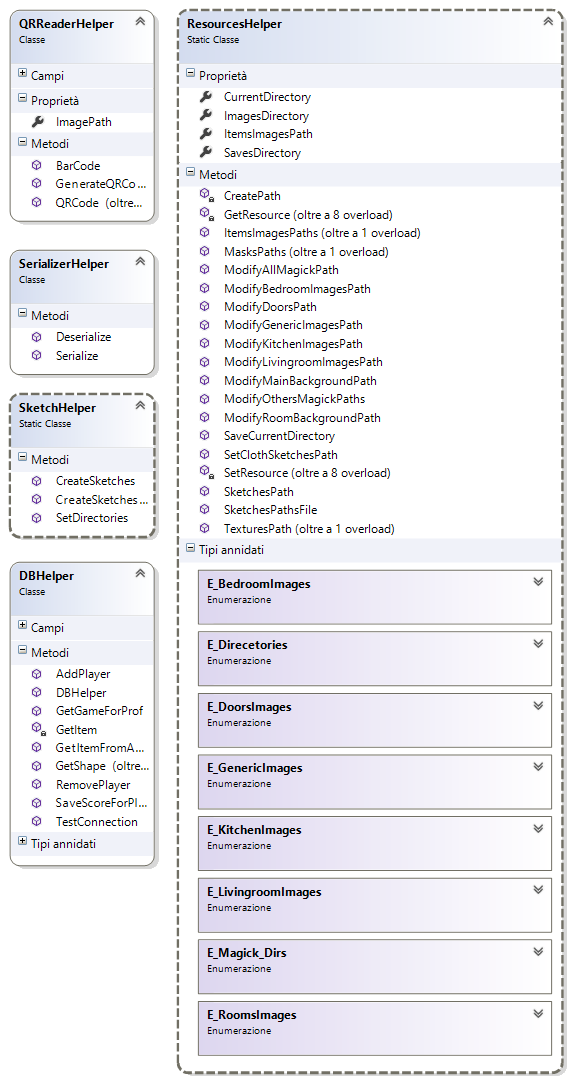
Acts as an interface with methods of ZXing, which are used to recognize BarCodes and QRcodes. We apply ZXing methods to pictures taken from the Kinect.

1. **Resource Helper**

It manages the App.xaml resources.

1. **Sketch Helper**

It’s static class that is used for the graphical interface. It uses methods provided by Magik.Net. It creates sketches of the Items that represent them in the game.

****

## **Controllers**

Controllers manages the game flow.

1. **PlayerChecker**

It checks who the player in front of the Kinect is. It’s used to manage the multiplayer features.

1. **KinectInterrogator**

Takes the images generated by the Kinect and passes them to the Bar- and QRCode readers already defined. It also check’s the BodyCount (how many shapes are in front of the Kinect).

1. **GameController**

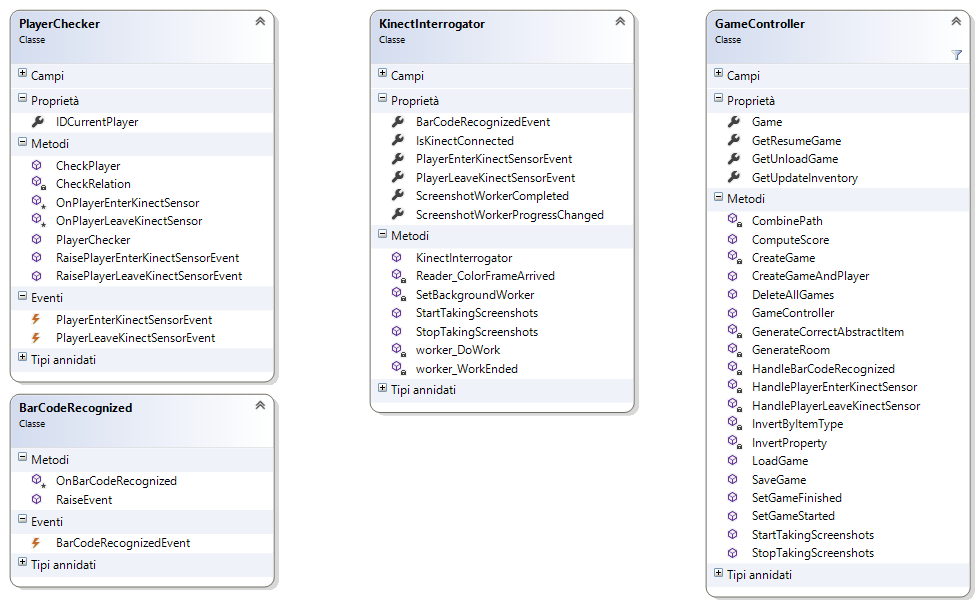
It creates the game and it manages the player context switch.

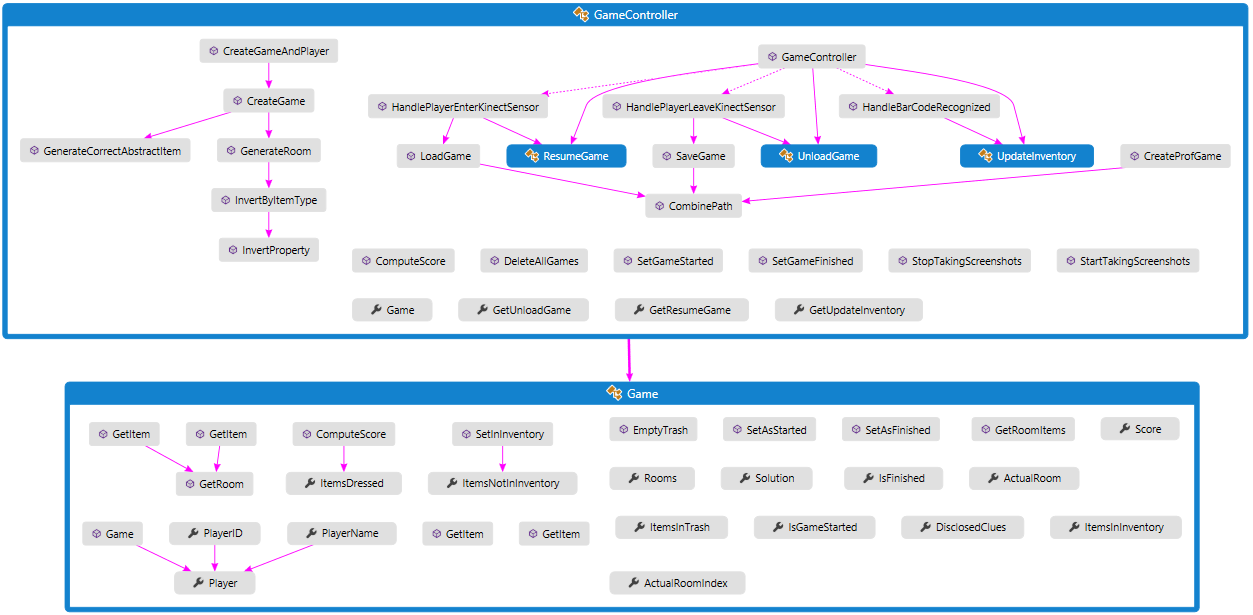
1. **BarCodeRecognized**

It’s a Dummy class that contains the event “Barcode recognized” which is triggered by KinectInterrogator and received by GameController

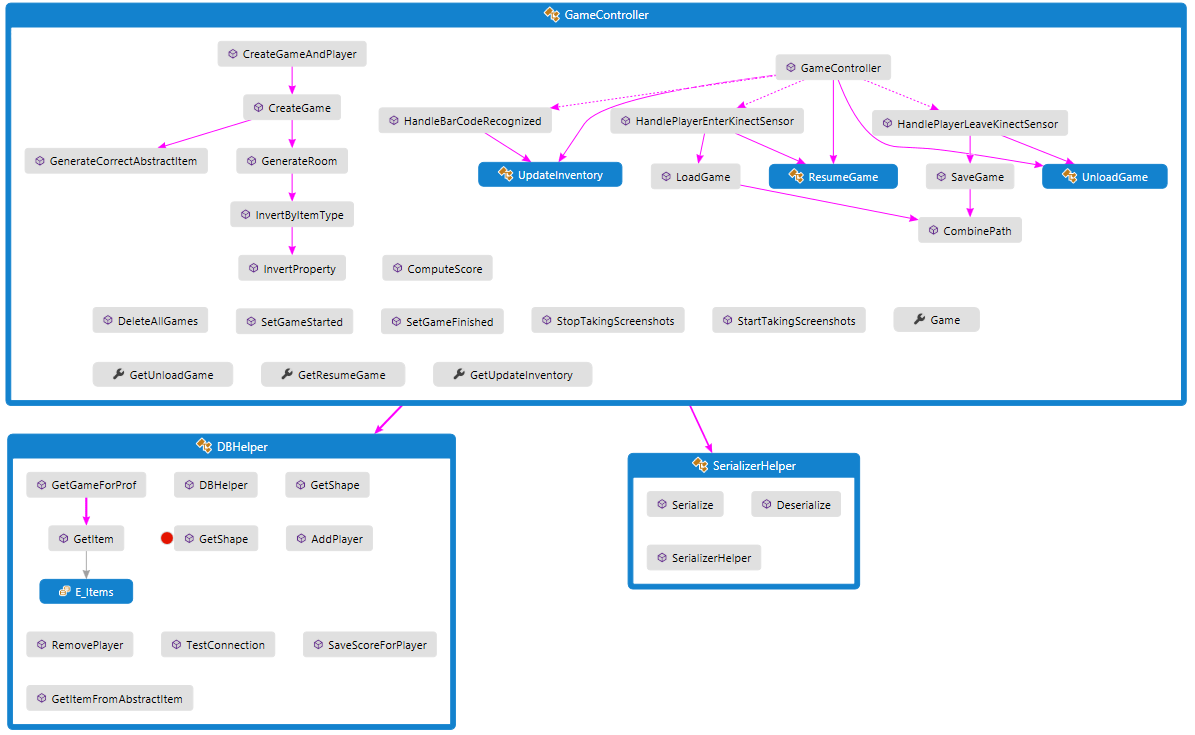
The game flow proceeds as follows:

KinectInterrogator takes the images from the Kinect. Then it send them to PalyerChecker whose job is to understand who is standing in front of the Kinect. If the shape corresponds to the player that is currently playing everything is fine, otherwise it sends a “player changed” notification of to the GameController. When GameController receives a “player changed” notification it saves the current context and notifies to the graphical component to unload the old game and wait for a new game to load.



****

Game controller -> model interactions

****

Game controller -> helper interactions

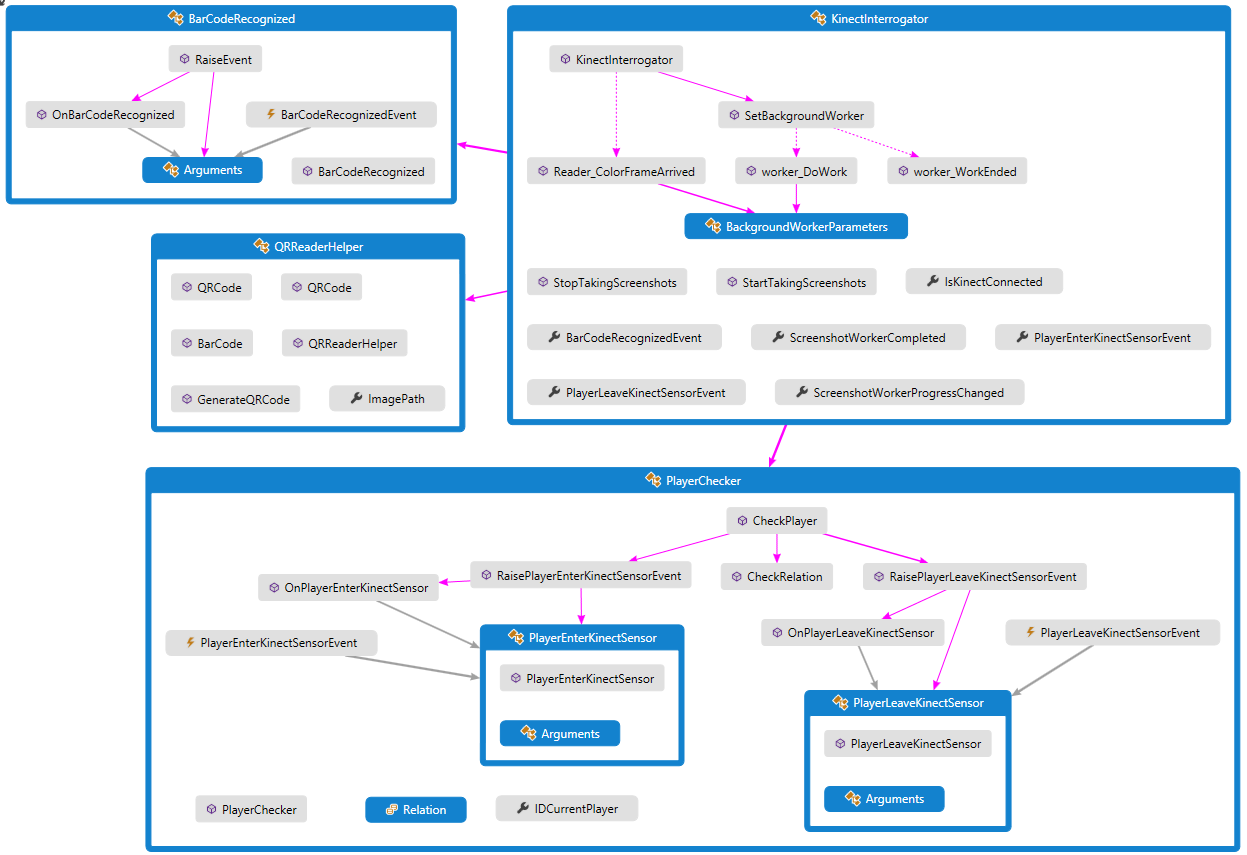
# **Interface**

Graphical choices :

* Color-changing door light
* Button placement

Graphic styles (.xaml)

The following page contains a schema describing the interactions between Kinect and Helpers

****

**Feasibility study**

**Technical Analysis**

This section explains the technologies adopted for the project and gives an idea of the required time

### **Hardware**

In order to realize the project the following hardware components are needed:

1. Kinect v2 sensor ,which includes a power hub and USB cabling
2. Desktop computer with the following characteristics:
   1. Operative system:
      1. Windows 8 (x64)
      2. Windows 8.1 (x64)
      3. Windows 8 Embedded Standard (x64)
      4. Windows 8.1 Embedded Standard (x64)
   2. 64 bit (x64) processor
   3. 4 GB Memory (or more)
   4. I7 3.1 GHz (or higher)
   5. Built-in USB 3.0 host controller
   6. DX11 capable graphics adapter
   7. At least 1 TB HD
3. A widescreen of at least 50’’

### **Software**

In order to be able to execute the project the following softwares are need:

1. One of the following Bar and QR code readers
   1. ClearImage
   2. BarcodeLib
   3. ZXing
2. Kinect drivers

### **Database**

The project extract its data from a database

It can extract data from an already existing DB (the one used by the shop) with small adaptation or a fully new database can be created.

The project can be adapted for each DBMS software which is already present

## **Timetable**

We can say that we were able to respect all the deadlines we put in our schedule

**Business Analysis**

This section analyses the project on the business point of view, so it concerns costs and estimated profits

### **Profits**

**Monetary profits**

After the installation of a system much simpler and less entertaining than the one here proposed a famous clothes shopping brand increases its selling by just less than 10% and the number of consumers augmented considerably.

We estimate an increase in selling of more than 10%

**Brand profits**

Installing and allowing the consumers to use a new technological entertainment system will also increase the popularity of the brand and of the shop

The more popular the shop is, the more people will come to visit it

**Costs**

1. Technology
   1. Hardware

|  |  |
| --- | --- |
| Kinect v2 | 150,00 € |
| Desktop PC | 500,00 € |
| Widescreen | 800,00 € |
| TOTAL | 1450,00 € |

* 1. Software

|  |  |
| --- | --- |
| ClearImage Licence | 800$ = 750€ |
| BarcodeLib Licence | 1599,00$ = 1500 € (unlimited) |
| ZXing License | Price to discuss with developers |
| Kinect drivers | Free |

Usually a shop has its own methods or licences to read BarCode, so the license can already be present. Only an adaptation process is needed

* 1. Database

The costs for the Database varies a lot.

If the DB is already existing the adaptation costs will depend on the original structure of the shop DB, so had to be discussed case by case.

If it had to be created ex-novo the price depends on which technology the client wants to adopt

1. Development works

This aspect covers the cost which concerns our own work and are relative to the times listed in the previous Gantt schema. Our request is 4500 €

1. Installation, configuration and maintenance

Our system is easy to set up and configure on the Hardware aspect, while it can take more effort in the DB aspect, since previous systems had to be integrated.

Configuration prices varies based on the complexity of the work required to personalize and adapt the system to the client needs; an average can be 600 €

The software is easy to maintain and update and the cost of it will be 200€/year

**TOTAL COSTS:** 7000 € +/- 10 % + IVA (VAT)

**TOTAL BENEFITS**: at least 10% augment of selling, augmented brand popularity

**NOTE: ricordarsi il discorso di TIM e delle applicazioni per la navigazione in negozio per invogliare gli utenti ad andare in un negozio fisico**

**Risk Analysis**

This sections explains which risks are take into account and which strategies had been adopted to avoid them

|  |  |  |  |
| --- | --- | --- | --- |
| RISK | RELEVANCE | PROBABILITY | COUNTERMUASURES |
| Data loss | Medium | Very low | No direct connection between DB and game - Secure DB queries |
| No item discovery | High | Low | Rules on item population |
| Repetitive Game | Low | Low | Randomization in the game |
| Difficult scaling | Medium | Average | Augment HW requirements |

The system is meant to be easy to use and intuitive. This because the game interaction through the Kinect sensor are similar to actions in real world.

The main risk lays in the configuration of the system in order to be adapted to previous solutions or to be personalized on client requests. This operation can be costly and can cause an augment of the price that can exceed the quote.

# **Conclusion**

## **Project decisions**

During the implementation we had to choose which values to give to many parameters. Here we show the reasons that lead us to the choice of the values for some of them

## **Main issues**

Even if the implementation of the project didn’t procced smoothly, we were able to address all the little problems that as usually appeared during the implementation phase with simple tricks that we already discussed before. The only real issue that we encountered was the image QR and Barcode recognition via Kinect. This caused a lot of trouble and delay and was partially solved with the application of some workstation constraint (listed in the next chapter).

To totally solve the recognition issue it’s enough to switch from a free recognition software to a licensed one (we give a full list in the Technical Analysis- Software paragraph). We did so with a time-limited trial copy and all the issues disappeared.

## **Workstation constraint**

These constraints are related to the recognition issue discussed above.

Put the Kinect at 1.40 m from the users in order to recognize the QRCodes.

The Kinect had to be placed at 1.30 m of height.

To read the barcode put it at 60 cm from the Kinect.

## **Further Works**

## **Why Choose US**

We offer a product which is simple and easily adaptable.

It’s simple under any aspect: easy to use, easy to play, has a simple and intuitive interface and it’s easy to configure.

Its easiness in configuration leads to its adaptability.

It’s adaptable because, as stated before, can be used on every existing DB

It’s also context adaptable.

We used a Fashion context because this was the assignment, but the game can be played and used in many different contexts. We can change shop nature (from fashion to toys, to hardware, to grocery) or even change the context from a shop to a waiting room to make the waiting less boring, to a kid’s space in a restaurant, to a museum to make the visit more funny and entertaining and so on.

All this factors made our project like a Swiss Army Knife: you bought just one instrument to solve many different issues.