

Developing a serious game as a tool for collecting data on food waste behavior

Christiaan Verloop

Supervisor
dr. A.M. Schaafstal

Critical observer
dr. J. Zwiers

Creative Technology, University of Twente

July 20, 2018

Abstract

A third of all produced food is wasted worldwide, much of which occurs at the consumer. Traditional research methods on food waste behavior have proven unreliable. Hence, new methods are being tested. The purpose of this study is to explore the possibilities of using a serious game as a tool for collecting data on food waste behavior. To this end, a game is devised, developed and evaluated to ultimately answer the questions; How can a serious game collect realistic data on food waste behavior? What data needs to be collected from the game, and how should the data be saved and formatted to be useful to the client?

The first chapter explains the origin and the context of the assignment. The second chapter explores the state of the art on serious games and data collection. In the third chapter, multiple ideas are devised after which one is chosen to be developed further. In the fourth chapter the chosen idea is specified, both in terms of gameplay, and in terms of data collection. Next is a description of the realization phase of this project, this includes the game, the data collection, and the documentation. In the next chapter, the game is evaluated with the use of two user tests. Finally, the paper concludes that the data collection and storage were a suitable solution for the developed game, however, it also concludes that the developed game is not a reliable tool for collecting realistic data on food waste behavior. This is followed by recommendations for the future to resolve the issues discovered in the developed game.

Acknowledgements

First off, I would like to thank Camille Gruter for her diligent work on the art side of this project. Next, I want to thank our supervisor Alma Schaafstal; throughout this project she has always had a clear vision of the goals of this project, she helped us stay on track when we risked straying off. Her critical view on the project has greatly increased the completeness of the end result. I'm also grateful to Anke Janssen and Rene de Wijk; not only did they provide us with an interesting and challenging assignment, they also took the time have regular meetings wherein we discussed and aligned our ideas with the goals of the project. Thanks also to our critical observer Job Zwiers for his feedback during the last stages of the project. Finally, I want to thank all the play testers of the game for their participation and feedback, it has helped tremendously in evaluating the game's success.

Table of contents

List of Figures	6
Introduction	8
Source of the assignment	8
Client goals and requirements	9
Scope	9
State of the art	10
Data storage	10
State based	10
Event based	10
Evaluation	10
Database type	11
Relational databases	11
Non-relational databases	12
Evaluation	12
Games with a similar topic	13
Food waste games	13
Cooking games	13
Evaluation	14
Ideation	15
Requirements	15
Brainstorm	15
Tamagotchi game	15
AR game	16
Restaurant game	16
Mother earth game	16
King game	16
Family game	17
Narrowing down	17
Specification	19
Game format	20
Data collection	21
Data format	22

Users	22
Savefiles	22
Events	23
DayStats	23
ProductInstances	23
ProductTypes	23
Settings	23
Realization	24
Game	24
Database	44
Documentation.....	47
Evaluation	48
User test 1	48
Results	48
User test 2	49
Results	49
Behavior.....	49
Data	52
Conclusion	54
Future work	56
Recommendations	56
Known bugs	57
Future possibilities	58
References	59
Appendices	60
Appendix A	60
Appendix B	62
Appendix C.....	63
Appendix D	64
Appendix E.....	74
Appendix F.....	77

List of Figures

Figure 1: Tamagotchi	15
Figure 2: Database overview	22
Figure 3: Login screen	24
Figure 4: Introduction screen	25
Figure 5: Dining room	25
Figure 6: Kitchen	26
Figure 7: Catapult	26
Figure 8: Recipe book	27
Figure 9: Statistics overview	27
Figure 10: Help screen	28
Figure 11: Village	28
Figure 12: Supermarket lane 1	29
Figure 13: Supermarket lane 2	29
Figure 14: Inside of a shelf	30
Figure 15: More product information	30
Figure 16: Basket	31
Figure 17: Supermarket cash register	31
Figure 18: Shopping bag	32
Figure 19: Street with greengrocer	32
Figure 20: Greengrocer	33
Figure 21: Greengrocer cash register	33
Figure 22: Street with butcher	34
Figure 23: Butcher	34
Figure 24: Butcher cash register	35
Figure 25: Castle entrance	35
Figure 26: Shopping bag in kitchen	36
Figure 27: Products in fridge	36
Figure 28: Recipe choice menu	37
Figure 29: More product information	37
Figure 30: Products on kitchen counter	38
Figure 31: Cooking part of the kitchen	38
Figure 32: Products in pans	39
Figure 33: King and guests eating	39
Figure 34: Finished dinner	40
Figure 35: Food waste in the catapult	40
Figure 36: Catapult being emptied	41
Figure 37: Food waste in village	41
Figure 38: More food waste in village	42
Figure 39: Decreasing satisfaction	42
Figure 40: Most food waste in village	43
Figure 41: Expired products	43
Figure 42: EventTable	44
Figure 43: DayTable	45
Figure 44: ProductInstances	45
Figure 45: ProductTypes	45

Figure 46: Savefiles.....	45
Figure 47: Users.....	46
Figure 48: Settings.....	46
Figure 49: 'Buitenbeentjes' - misshaped fruits and vegetables sold by Albert Heijn.....	58
Table 1: Unnormalized database.	11
Table 2: Normalized database.	11
Table 3: Normalized database.	11
Graph 1: Average number of shop visits per day.....	50
Graph 2: Time until dinner was served.	50
Graph 3: Grams served per number of eaters	51

Introduction

A third of all produced food is wasted worldwide, much of which occurs at the consumer (FAO, 2011). Much research has been done as to what foods are wasted, and how to decrease the amount of wasted food. One such research institution is the Wageningen University & Research.

Source of the assignment

The department Food and Bio-based Research at the Wageningen University & Research conducts extensive research on food waste. One of their projects is 'Houdbaarheid begrepen' (Wageningen University & Research, 2016), which focusses on expiration-date related food waste. The goal of this project is to reduce spoilage in shops and at the consumer. To achieve this goal, new interventions need to be developed to decrease food waste in households. Additionally, people need to be educated about the possibilities of saving leftovers, and the meaning of TGT/THT dates.

Before interventions can be developed, it needs to be known what kinds of products are currently being wasted, when they are wasted and why they are wasted. There are several methods for acquiring such information. One of the most common ways is through surveys. These are quick to perform and easy to distribute across a large amount of people. They are also very cheap. The main downside is that the answers given by the participants are self-reported. This can be problem depending on the subject of the survey. When asked about their behavior, people can feel embarrassed, or they can feel like they have to conform with societal norms. This was found to be the case when using surveys to gather data on food waste behavior. People tended to give socially desired answers, rather than accurately reporting their food waste behavior. In some cases, the survey can also make people aware of their food waste behavior, which can cause a desire in the participants to improve their behavior. This may sound like a good thing, however, people tended to fill in the survey as if they had already improved their behavior. While in reality, they often forget about their good intentions within the week.

For these reasons, surveys have been found to be unreliable for gathering data on food waste behavior. That's why other methods have been considered. One of these is to make people keep a diary in which they write what products they buy, what the expiration date was, where they stored the product, when they used the product, and whether they wasted the product. This is a lot of effort on the participants side, which means that a monetary incentive needs to be offered. This makes this type of research very costly, especially since such a diary would need to span several weeks. On top of that, the main problem of surveys is also present in this method; the behavior is self-reported, so people can sugarcoat the truth.

Another way to gather data about food was is to dig through garbage bins. This is dirty work, but it avoids the problem of self-reported behavior. This method can give insight as to what products are wasted most, however, it doesn't reveal much information about when the products were wasted, where they were stored, and why they were wasted.

Another way is to follow people in real life to observe their behavior directly, however, this is even more expensive than keeping diaries and comes with big privacy concerns. Additionally, people are likely to behave differently when they feel watched.

Since none of the aforementioned methods suffice for collecting data on food waste behavior, a new method had to be devised. The project leader Anke Janssen, and two of her colleagues Rene de Wijk and Hilke Bos-Brouwers came up with a new idea: To put participants in a simulated environment in which they perform all the actions that make up the food cycle in households. This simulation could be in the format of a serious game. The game then automatically collects data on people's actions. This idea eliminates much of cost- and privacy-concerns of the other ideas. Additionally, the behavior is measured unobtrusively, which means people don't feel constantly watched. The researchers (from now on 'the client') contacted the University of Twente to see if two students could explore the possibilities of such a game.

Client goals and requirements

The client defined two goals for the serious game. The first and primary goal is to collect realistic data on food waste behavior. The secondary goal is to improve people's food waste behavior. The target audience for the game are people involved in the provisioning, storing, preparation, consumption and wasting of food in a household. The client is mostly interested in the behavior of people who go grocery shopping once or twice a week. Additionally, the data needs to be collected remotely.

Scope

This game will be made by Camille Gruter and Christiaan Verloop for their graduation project. The timespan of this project is 5 months, half of which will be used to come up with a clear vision for the game. The other half will be spent on creating the prototype and evaluating it. In this project, Camille will focus on the art and design of the game, while Christiaan will be responsible for programming of the game and the data collection. Therefore, this paper will focus on the programming and data collection of the game.

During the development of this game, both students will try to answer the question 'How can a serious game collect realistic data on food waste behavior?' The prototype that will be developed serves as basis to provide answer to this question. Furthermore, this paper also aims to answer the question of what data need to be collected from the game, and how should the data be saved and formatted to be useful to the client.

State of the art

With the aforementioned goals and requirements in mind, the next step is to look at what already exists in terms of data storage, and games in general. The first section will describe the different ways in which game data can be stored. The second section will look at the type of database the data could be stored in. Finally, the last section will investigate existing games with a similar topic to the one that will be developed.

Data storage

This section will look into the different ways of storing game data, to find out what methods are most suitable for this project. In general, there seem to be three main ways of storing data; state-based, event-based, and artifact-based (Allen & March, 2006). Only the former two will be described in more detail below, since the later has grown out of fashion.

State based

When a change happens to an object in the game, the game stores the new state of this object in the database. For example, if a player had 50 euro, and bought something for 20 euro, the new entry in the database will show 30 euro, which is the current balance of the player. This type of data storage makes it easy to see what the current state of the game is. It does not show however, what the 20 euro was lost, or where it was spent on. Therefore, this type of data is most useful in situations where variables can only change for one reason, for example, in a game like snake, your length can only increase by eating a block. In the money example however, the money could have been spent on many different things. These details are lost in state-based data storage.

Event based

In event-based storage, the game saves events, rather than states. So, when looking at the same money example, the new entry in the database will say -20 instead of 30. These events are stored in an event table and allow more detail about the properties of the event to be saved. For example, there could be a column specifying what type of event happened, in this case a purchase event. Furthermore, an item ID could be specified to see what item was purchased. The main downside to event-based data storage is that it is harder to see the current state of the game. To find this out, the starting state needs to be known, after which all the events need to be reconstructed. Nevertheless, people familiar with queries are on average more accurate in formulating queries in event-based data representations than in state-based representations (Allen & March, 2006).

Evaluation

Both state-based and event-based data storage have their advantages and disadvantages. For this project however, it is very important to know what actions lead to a result. For this reason, event-based data seems most appropriate. The downside of having to reconstruct all actions to retrieve a variable at specific time could be mitigated by combining the two; in addition to storing the change, the new value could also be stored. This would make the table very superfluous however, so a possible solution is to only store the new value infrequently, for example at the end of the day, since for most values, it doesn't need to be known what exactly it is at all times.

Database type

Once data has been collected, it needs to be stored. This section will explore the various ways of doing this. Nowadays there are two main categories of database systems; relational, and non-relational. Each come with their advantages and disadvantages.

Relational databases

Relational databases have been the standard for many decades now. They often consist of multiple tables with a fixed number of columns and a variable number of rows. Within a row, there can be information about different properties (the columns), or there can be a reference ID. This reference ID can be looked up in a different table to view more information about it. A relational database that doesn't follow this pattern is not normalized. An example can be seen in table 1. This table stores the customer information in every purchase. This creates a lot of duplicate data. Instead, the table can be normalized as seen in graph 2 and 3. Each purchase now has a reference ID, which can be looked up in another table to view more information about the customer.

Item	Price	Amount	Date	Time	Customer name	Customer phone number	Customer email
Apple	0.4	1	7/7/2018	1:23 PM	John	612345678	John@gmail.com
Banana	0.32	2	10/7/2018	10:28 AM	Peter	687654321	Peter@gmail.com
Peach	0.5	4	20/7/2018	3:13 PM	John	612345678	John@gmail.com
Pear	0.22	3	20/7/2018	3:15 PM	John	612345678	John@gmail.com

Table 1: Unnormalized database.

Items	Price	Amount	Date	Time	Customer ID
Apple	0.4	1	7/7/2018	1:23 PM	1
Banana	0.32	2	10/7/2018	10:28 AM	2
Peach	0.5	4	20/7/2018	3:13 PM	1
Pear	0.22	3	20/7/2018	3:15 PM	1

Table 2: Normalized database.

Customer ID	name	phone number	email
1	John	612345678	John@gmail.com
2	Peter	687654321	Peter@gmail.com

Table 3: Normalized database.

Relational databases have a predefined structure; there is a fixed number of columns, each of which can hold a predefined data type. Because of this, the database it is known exactly what data is held in each column. The predefined structure allows for complex actions, like joining tables bases on values in certain columns. Tables can even be rolled back to a previous state. The downside to the predefined structure however, is that data with a different, or no structure cannot be saved in a relational database. By far the most popular relational database management systems are Oracle, MySQL, and Microsoft SQL Server (Solid IT, 2018).

Non-relational databases

Before relational databases became the standard, non-relational databases were used. However, with the introduction of relational databases, these quickly became irrelevant for the majority of tasks. Recently, they have been making a comeback however (Solid IT, 2018). This is because certain applications, particularly ones with a high data velocity, unknown, or rapidly changing data structure cannot be handled well using relational databases. Furthermore, relational databases tend to slow down when dealing with huge volumes of data. (Győrödi, Győrödi, Pecherle, & Olah, 2015) Non-relational databases are able to store semi-structured and even unstructured data. This gives great possibilities, but it means that the database will not verify if the structure of the data is correct. This means that the application which reads and writes to the database is responsible for delivering data in a useable format. Non-relational databases are generally faster for extremely big databases. This is because they do not support computationally heavy functionalities like joining tables and rolling back tables to a previous state. There are different types of non-relational database, three of which are outlined below (Győrödi, Győrödi, Pecherle, & Olah, 2015).

One of the simplest types of non-relational databases is the key-value store. This database can only store pairs of keys and values. When a key is known, a value can be retrieved. Values can either be all of the same type, in which case no extra type information needs to be attached, or of different types, in which case each value comes with type information. The simplicity of key-value stores is often inadequate for complex applications, however, the simplicity also allows for very high performance. The most notable example of a key-value store is Redis.

Another type of non-relational database is the wide column store. This type of database does not have a fixed number of columns, in fact, the data type of a column can be different for each row. This brings great flexibility. For example, graphs 1-3 currently only support one phone number. A wide column store however, allows you to add multiple columns, to store multiple phone numbers per customer. The most popular examples of wide column stores are Cassandra and HBase.

Document oriented databases are the most flexible of the three. These store information similarly to key-value stores. They store a key, but instead of a simple value, they store a document as a value. This document can consist of multiple other values of various types. Documents can even have other documents nested within them. The most popular example of this type of database is MongoDB.

Evaluation

For this project, a relational database seems most suitable, since the structure of the data will be easy to predefine, and the size of the database will reach nowhere near the amounts at which slowdowns occur. Furthermore, relational databases are more well-known, which helps the client in using the system. Of the most popular relational database management systems, MySQL is the most accessible since it has an open source license, and it runs on virtually all platforms and is thoroughly documented.

Games with a similar topic

Before starting development, it's important to know what games with similar topics already exist, to find out what parts may be relevant for this project.

Food waste games

There are several 'games' about food waste, however all of these are closer to quizzes than actual games. They test your knowledge and bring awareness about food waste. Because of this they have a very low replay value.

'Food savers' is an online board game developed by the project 'Food Waste Effect'. (Food Waste Effect, 2017) Players all start at the beginning of a path consisting of 64 tiles. The goal of the players is to be the first one to reach the end of the path. To do this, players have to answer questions on each tile they land on. If a question is answered correctly, the player can take a few steps. The path goes through four different areas each of which has questions about a different topic related to food waste. The game was made to educate people about food products, processes, traditions, food poverty and food waste. The target audience for this game is families, so both adults and kids.

'Zero waste' is a game developed to teach kids about food waste and recycling (Jim Metzner Productions, 2014). This game takes place in a dirty city. The city consists of 10 areas each with a question related to waste or recycling. Every correct answer makes that area of the city cleaner. The goal is to get as many questions correct as you can. The target audience for this game is kids from primary schools.

'Rethink waste' is a game developed by the city of Surrey to educate its inhabitants about recycling (City of Surrey, 2018). The player has to sort all kinds of waste (some of which is food waste) into 4 different bins. There are 5 levels, each of which has six items to sort. If the player puts an item in the wrong bin, the player can retry without any penalty. Once a level has been completed, the player can add a decoration to their park. The more decorations in the park, the more visitors it attracts.

Cooking games

Since not many games are about food waste specifically, cooking games were also looked into, since they have an overlapping element with food waste. The following games were evaluated to see whether they provide relevant aspects to the subject of food waste.

- Overcooked (Ghost town games, 2018)
- Order up!! (SVS Games, 2008)
- Cooking dash (Glu games, 2016)
- Yummy Yummy Cooking Jam (Virtual toys, 2008)
- Food Truck Chef (Nukebox studios, 2017)
- Cooking Joy (Google Play, 2017)
- Cooking fever (Nordcurrent, 2014)
- Cooking Mama 5 (Office Create, 2013)

The games won't be described individually, since all but the last in the list have the same core game mechanic; the player is the chef of a restaurant, and needs to fulfill customer orders. The player has an unlimited resource of ingredients, which need to be cut, put in a pan or on a grill, and put on a plate, after which it can be given to the customer who pays money for the meals. These games usually

have levels which have a target amount of money that the player needs to reach to complete the level. These games are usually targeted towards kids, mostly girls.

Only two of these games stand out in terms of having added mechanics. Overcooked is the only multiplayer game in the list. Players need to cooperate to finish the meals in time. In contrast to all the other games, this is not a point and click game, but one where the player moves around a character. This, combined with the multiplayer aspect can make this game quite chaotic. The developer embraces this and further amplifies the chaos by having different levels with different challenges, such as splitting the kitchen up into difficult to cross parts. This makes for an engaging couch experience that even many adults enjoy.

The other game that stands out is Cooking Mama 5. Unlike all other games, this game does not focus on customers, players can just make meals that they want to make. The cooking in this game is probably the most detailed of all the games in this list. Additionally, the game also has different modes, one of which is a minigame wherein the player must harvest food from trees, or catch food fallen off a truck. In another mini game, the player has to try and fit all kinds of products into the fridge.

Evaluation

The games about food waste discussed in this section were made to create awareness about food waste. All of them do so by testing your knowledge. This can be an effective way of creating awareness, however, no real-life or simulated behavior is measured.

The cooking games discussed do simulate a part of the food cycle, so behavior can be measured. Most of the games listed take place in a restaurant. This setting is different from the client's needs, since food waste is handled differently in restaurants than in households. Furthermore, all but one game don't include the provisioning and storing of products, and none of the games take due dates and food waste into account. The game that stood out most is Cooking Mama 5, since it takes place in a household, rather than in a restaurant. Furthermore, this game includes 3 aspects of the food cycle in households; provisioning, storing and preparing.

The food waste games don't measure behavior but quiz the player about food waste. This can help in improving people's food waste behavior but will not help in collected behavioral data on food waste. Inspiration can be taken from the preparation aspect present in all the cooking games. Additionally, the storing aspect from Cooking Mama 5 could be relevant for this project too. The provisioning aspect of Cooking Mama 5 is less relevant, since the food isn't bought.

Ideation

Requirements

The client had specified some initial goals and requirements for the game, but before ideas can be generated, more in depth information about the requirements needs to be acquired. After consulting the client, it became clear that the game must function as a research tool. This implies that the client can tweak the input variables to see what the effects are on people's behavior. This is necessary to gain insight in the questions the client wants to discover:

- Are people money driven, health driven, or convenience driven?
- How much do these factors (money, health, convenience) affect food waste?
- In which steps of the food cycle in households is food wasted, and why?

In order to acquire accurate data for these questions, it is important that the action in the game are similar to the actions in the real world. In other words: if a person would do a certain action in the real world, the person should also be tempted to perform the same action in the game. This means that all the actions in the food cycle in households must be present in the game: provisioning, storing, preparation, consumption and wasting. Furthermore, aspects that play a role in these actions should be present in this game.

The secondary goal of the game is to improve people's food waste behavior. To achieve this, players need to be made aware of the food waste they make, and they need to feel the downsides of this.

The client further specified that the game needs to be in Dutch. This is because the target audience will be exclusively Dutch.

Brainstorm

With the requirements in mind, many ideas were conceived. All of them include a way for the player to see the impact their food waste is making. This is to satisfy the secondary goal. To satisfy the primary goal, all the games contain the steps of the food cycle in households.

Tamagotchi game

One of the most abstract ideas was a Tamagotchi-like game wherein the player needs to feed a critter by buying food and giving it to the critter. If too much food is given, the critter refuses to eat it and the food would drop down to the ground. This food waste piles up until it engulfs the critter at which point the player loses the game.

This game didn't focus enough on expiration date, and the purchasing and storing was too far removed from reality. Additionally, the food waste was too visible. It must be visible to satisfy the secondary goal of the game, but if the food waste is too visible, it can affect the data. For these reasons, this idea was discarded.



Figure 1: Tamagotchi

AR game

The most true-to-life game was one wherein the user uses the camera of their mobile phone to take a picture of the products in their shopping cart. Then, a puzzle game would be generated based on that photo. Similarly, for the other stages in the foodcycle (storing, preparing, consuming, wasting), another photo is taken, and a new puzzle is generated based on it. The photos would give great insight into what people buy, use and waste, however, some crucial elements, like expiration date are missing in the photos. Furthermore, analyzing photos to recognize products is time consuming work. This could be automated by the game using artificial intelligence, however, after looking into the feasibility of this, it was quickly found that it would be very technologically challenging. Additionally, players' privacy is at risk in such a game, because there could be sensitive information in the pictures taken, either in the background, or in the products themselves.

Restaurant game

Another idea was a game wherein the player manages a restaurant. The player must buy, store, and prepare food, and any food not eaten is thrown away. Guests would reserve tables in advance to allow the player to buy enough food. Any food wasted in the restaurant would pile up either within the restaurant, or near the entrance. This is to make people more aware of their food waste.

This idea was quickly abandoned, since food waste is handled very differently in restaurants than in households, because of the large number of eaters, the freshness that customers demand, and the fact that you need to have all ingredients for all recipes in stock, because you don't know what customers will chose. For these reasons, the restaurant seemed least appropriate.

Mother earth game

Focusing more on small-scale cooking, was the 'mother earth game'. This game takes place in a small village. In the middle of this village is a mythical being called 'mother earth' who protects and conserves the village. 'Mother earth' receives money from donations of the village. The player is allowed to use this money to feed 'mother earth' by buying food in the village, storing it, preparing it, and serving it to 'mother earth'. Any food waste gathers around mother earth.

The mother earth idea had a mythical vibe to it, which can be an interesting style for a game, but the mythical nature also caused some mental steps and illogicalities, like 'How exactly does mother earth protect the village, and from what?' and 'If mother earth is so important, why does the food waste gather around her?' These mental steps remove the player further from the real world, rather than staying close to it, this is likely to cause people to behave less like in the real world, since the setting is alien to them.

King game

The king game is similar to the 'mother earth game' idea, except it takes place in a world more like the real one. The player is the cook of a king. The king gives the player a weekly budget which the player can use to buy food in the village of the king. The player can then store the food, prepare the food, and serve the food to the king. Any food that is thrown away is catapulted into the village. The objective of the player is to keep both the king happy by making good dishes and to keep the village happy by not catapulting too much food into the village.

This idea stays closer to the real world than the 'mother earth game', however, there are still some peculiarities, for example, a king ruling over just one village, however, the mental translation from a

village to the country is easily made. There is still an element of fun in the game while staying close to the real world.

Family game

The family game takes place in a world much like ours. The player is responsible for the food of their family. The player must buy food, store food and prepare food for their family. The food is then eaten by the family. Any leftover food is throw away. This idea stays closest to the real world, however, it also is the least appealing as a game, since people already do these activities for their families in real life. This idea was discarded because it was considered 'too boring'.

Narrowing down

After all ideas were considered, the king game was chosen to be developed further because it stays close to the real world, while staying fun and interesting as a game. To further define and shape the game, a list of 100 factors that affect the actions in the food cycle was conceived. This list can be found in appendix A. Together with the client, a selection of factors was made that were interesting and relevant for the game. These factors were then integrated into the game as follows.

- Price – Products have a price reflective of real life. Products in the fresh stores are more expensive than in the supermarket.
- Quality – All products are of high quality, but products in the fresh stores are slightly better.
- Health – Will not be implemented in detail yet, but products from the fresh stores are healthier.
- Freshness – Products from the fresh stores are fresher than products from the supermarket.
- Use-by-date – In the shop, products have a predetermined due date. This due date indicates how many more days it can be used if the product is stored in its current location. For example, an apple might have a due date of 5 days uncooled. If it is placed in the fridge however, the due date is higher. Products that have a due date below zero are expired and no longer taste good.
- What do I still have? – This factor is implemented by having storages in the kitchen; cupboard, fridge and freezer.
- Number of eaters – The king's family will often join him, however, not everyone is always present. The number of eaters each week is constant, however, the distribution per day can be different. This is to keep it fair to the player, who receives the same budget every week.
- Money Available – At the start of each week, the player gets a predetermined budget.
- Portion size – Each product in the shops has a specified number of grams. For this version, no different portion sizes are available for the same product. However, the engine will allow it.
- Packaging exterior – Each product in the shops has a picture of the product. For this version, no variations in packaging exterior are available for the same product. However, the engine will allow it.
- 35% discount sticker – For this version no discounted products are available, however, the engine will allow it.
- Distance from shop to home – The supermarket is on the village square, which is closest to the castle, the fresh stores are in the streets adjacent to the village square and require an extra click to get to.
- When should dinner be ready? – The player has a time limit each day. If it's already late on the day, the player might not have time to go shopping, and make dinner in time.

- Storage size at home – The storages in the kitchen have limited capacity. If a storage is full, a product either needs to be moved to another storage, used, or thrown away.
- Planning – The player can choose a recipe from the recipe book and go shopping to get those ingredients. If the player did not plan ahead, they might forget a product. Planning can also be used to save time, for example, by shopping for multiple days on one day, the player will have enough time to prepare a good meal the next few days.
- How much of each storage type do I have? – Each storage has its own capacity. If the freezer is full, a product can be moved to the fridge, however, this does impact the due date.
- Use, move or waste? – If a storage is full, the player can choose to use a product, move it to another storage, or waste it.
- Is product expired? – If a product has expired, the player can throw it away, or use it, but it will not taste good anymore.
- What do I want? – The player is free to choose what recipe they want to make. The king has no preference.
- What do I have? – Before cooking, it is wise to check if the player has all the products required for the meal.
- Number of eaters – The amount of food to prepare depends on the number of eaters. If too much is made, the remains are wasted, if too little is made, the king will be dissatisfied.
- How much do eaters eat? – Each eater has a minimum and maximum amount of food that will satisfy them.
- I'm full – If an eater gets too much food, the remains are not eaten. These leftovers need to be thrown away.
- The food wasn't prepared well – If ingredients are raw or burnt the eaters will not eat the food. The remains need to be thrown away.

After these factors were integrated into the game, a preliminary chart was made to confirm with the clients that the idea was clear to everyone, see appendix B. This chart shows all the different scenes in the game and the properties of each scene.

Specification

The next step was to prepare work on the prototype. For the development part of this project, this meant making a class diagram showing all the properties and interactions of objects in the game, see appendix C. This class diagram was discussed with the client to make sure the idea of the game aligned with the clients' needs. After further discussion with the clients, the game and the data collection were specified as follows.

- The player is responsible of making dinner for the king and his guests.
- The player can buy food in the shops in the village.
- The village contains a supermarket and fresh stores.
- The supermarket is more easily accessible than the fresh stores.
- Products in the supermarket are cheaper than products in the fresh stores.
- Products in the fresh stores are tastier than products in the supermarket.
- All shops are open every day of the week.
- The prices of the products should be in euros and should reflect real life prices.
- The use-by-date of a product is expressed in days left.
- The use-by-date is always the same for a certain product in the shop.
- The use-by-date should reflect real life use-by-date.
- Products should be dry, cooled, or frozen, reflective of real life.
- The use-by-date of a product is based on the type of storage it is in (dry, cooled, frozen).
- There is a limit on the number of products the player can carry.
- The player cannot spend more money than they have.
- There is time limit each day, which indicates when the king wants to eat.
- Bought products must be stored in one of the storages of the kitchen.
- The kitchen has dry, cooled, and frozen storages, each with their own capacity.
- The use-by-date of a product increases if it is stored colder than in the supermarket.
- The use-by-date of a product decreases if it is stored warmer than in the supermarket.
- The use-by-date of a product decreases by one day per in-game day.
- If a product has expired, it is made visually clear that this is the case.
- Products in the storages can be thrown away at all times.
- The player has recipes available in the form of a recipe book.
- The recipe specifies the amounts for each ingredient for a 4-person dish.
- For each ingredient, the player can choose how many grams to add.
- If a recipe is followed properly, the dish tastes good.
- If an ingredient is missing, or was added at the wrong time, the dish tastes less good.
- If one or more expired ingredients were used, the dish tastes less good.
- If a dish is prepared for too short, or too long, the dish tastes less good.
- A bar above the pan indicates the progress of the dish.
- The king is satisfied when a dish tastes good, if it is enough to feed all eaters, and is in time.
- All eaters have the same minimum and maximum required amount of food per dish.
- If the king is satisfied about a dish, the player receives bonus points.
- If too much food is prepared for the number of eaters, the extra food is to be thrown away.
- Food that is to be thrown away is loaded into a catapult.

- The catapult costs money to fire.
- Food waste that has been catapulted lands in the village.
- The food waste in the village is made visual.
- Food waste is measured in grams.
- The more food waste in the village, the less satisfied the village is.
- At the start of each in-game week, it is made clear how many eaters there will be on each day.
- The number of eaters each week is predefined.
- The distribution of eaters per day can vary between weeks.
- At the start of each in-game week, the player receives a predefined budget.
- At the start of each in-game week, money not spent in the previous week is lost.
- Bonus points can be gained by making good meals for the king.
- Bonus points can be used to unlock new recipes and ingredients.
- The skill level of the player is determined by the satisfaction of the king + the satisfaction of the village.
- The client needs to be able to easily tweak the following values:
 - Capacity of each storage
 - Weekly budget of the player
 - Number of eaters per week
 - Cost of the catapult
 - Scale factor for the village food waste
 - Duration of a day
 - Penalties and bonuses in a dish
 - Minimum and maximum amount of food eaten per guest
- Products and recipes need to be able to be added and edited without having to change the game code.

Game format

The game will be a 2D game. This was chosen because the target audience is more familiar with 2D games than 3D games, and because the addition of a third dimension does not add much value in terms of gameplay or data collection.

The client indicated no preference concerning the platform the game would be developed for (PC, mobile or web). Since web and mobile are more accessible for the target audience of this game, web was chosen, because it requires no downloading and installing, which makes it easier to get started. Furthermore, no information is saved on the player's device.

Data collection

The client indicated that the following data were important to be collected:

- What has been bought + expiration date
- What is used + expiration data
- What is not used and when is it thrown away + expiration date
- How much money is spent per shop visit and per week
- How much food is prepared (for how many people)
- Was the recipe followed? Are all ingredients present?
- Is the preparation time correct, too short or too long?
- How often is the catapult used?

The data required to answer these questions could be saved literally, which would make it very easy to retrieve the information. However, this would result in a very unorganized and superfluous database. For this reason, the use of a relational database was proposed. The retrieval of data in a relational database requires queries to gather information from multiple tables at the same time. The client was comfortable enough with this to agree on the usage of a relational database. The following structure was conceived, note that not all the data that answer the questions of the client are saved literally; some things need to be derived.

- All properties of products, including:
 - Name
 - Grams
 - Price
 - Due date
 - Shop
- Each interaction with a product and the expiry date at the time of interacting, this includes:
 - Moving
 - Using
 - Wasting
- An overview of statistics per day, this includes:
 - Properties of the meal
 - Grams served
 - Was the food still raw
 - Was the food burnt
 - Was an ingredient missing
 - Was an expired product used
 - Was the meal served on time
 - Satisfaction of king and village
 - Food waste made
 - Number of eaters
 - Money left
- Miscellaneous actions like:
 - Using the catapult
 - Visiting a shop
 - Serving dinner

Data format

The client has indicated that the data should be readable and processable by Excel or SPSS. The data will be stored on the server in an SQL database, where the tables can be exported to a CSV format (Comma Separated Values). This format is supported by both Excel and SPSS. The database will use the following seven tables.

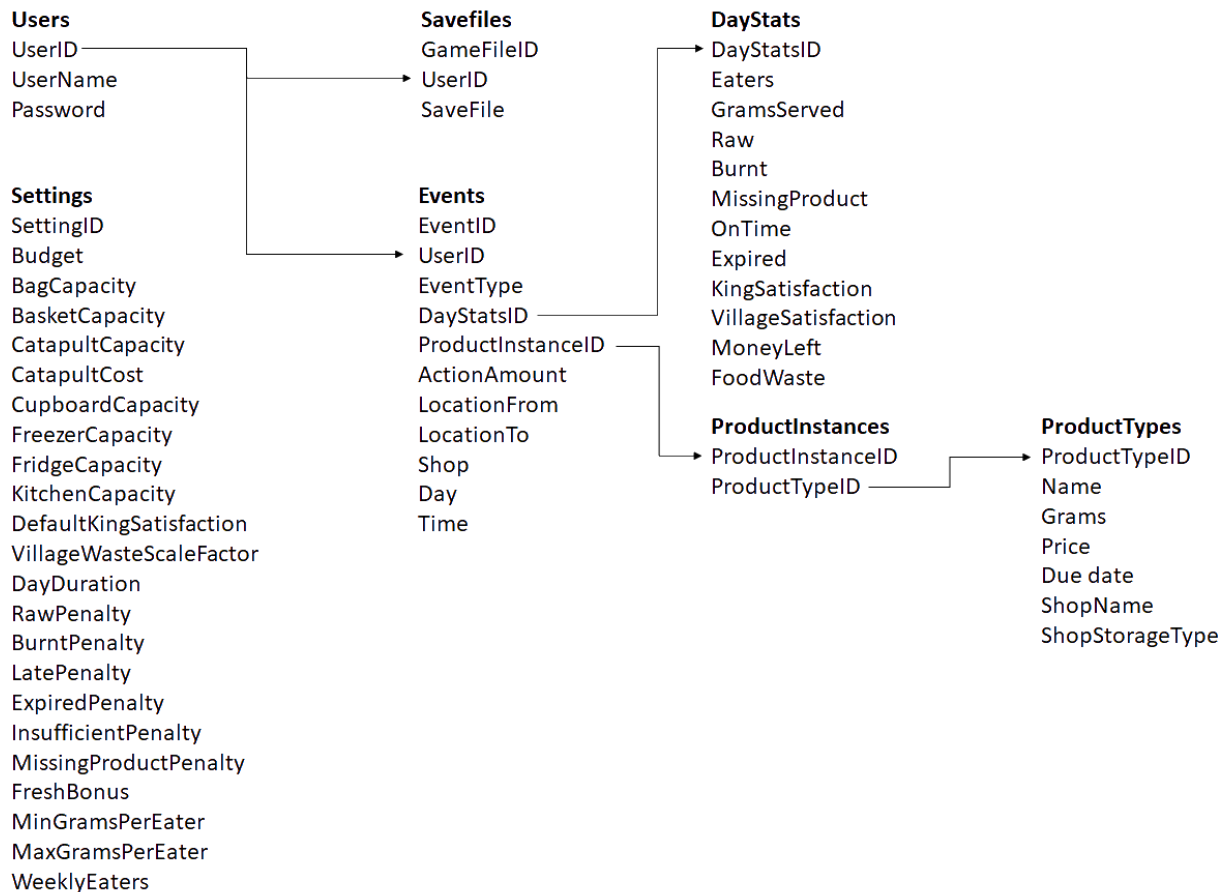


Figure 2: Database overview

Users

The users table stores a user ID, a username and a password. The user ID functions a unique identifier for each player. (Currently the system does not allow users with the same name to register, so the username could function as a unique identifier as well, however, in the future the game could use email-based registration which could allow non-unique usernames.) This ID can be used to look up data about the player in other tables. The username and password are used to login to the game. The passwords are currently stored in plain text. This is not an appropriate solution for a public game, since all users could lose their accounts if the database got hacked. The game is currently still closed to the public, and the clients will only ask small groups of people to play the game for their first tests, so non-encrypted passwords suffice for now.

Savefiles

The saveFiles table stores the progress of the players, this way, players can log out of the game, and come back later to continue where they left off. When a user logs in, the system loads the most recent saveFile of the player by selecting the saveFile with the highest GameFileID that matches the UserID.

The SaveFile is a JSON string which stores all the variables that determine the current state of the game. This table is not relevant for the client but is necessary to save player's progress between play sessions.

Events

The events table stores all relevant events for the client. This includes all interactions with a product, shooting the catapult, visiting a shop, and ending the day. Each event is signified by an EventID. The EventType indicates what type of event happened. The DayStatsID column is only relevant in the case of an 'endDay' event. It is a reference to the DayStats table which saves the details of the day that just ended. ProductInstanceID is used to signify which productInstance a certain event happened to. Examples of such events are 'using a product', 'moving a product', and 'wasting a product'. The ActionAmount indicates how much of a product was moved, used, or wasted. LocationFrom saves where a product was moved or wasted from, while LocationTo saved where a product was moved to. Shop saves which shop was visited in the case of a shopVisit event. Each event is accompanied by a Day and Time. Days are measured in terms of in-game days, while Time is measured in frames.

DayStats

The DayStats table stores the properties of a certain day. This includes the number of eaters that day, how much food was served, and was there anything wrong with the meal? E.g. Raw, burnt, late, expired, missing product. Furthermore, it stores the satisfaction of the king and the village at the end of the day. Lastly, it stores how much money the player has left, and how much food waste has been made.

ProductInstances

This table exists to look up what type of product a certain product instance is. The reason a distinction needs to be made between productTypes and productInstances is because there can be many instances of a certain product type. For this reason, the ProductInstances table has a ProductInstanceID that matches with a ProductTypeID. This ProductTypeID can then be looked up in the ProductTypes table to view more details.

ProductTypes

The productTypes table stores all information about products. This includes the name, the amount, the price, the due date, the store it is sold in, and the storage type it is sold in (dry, cold or frozen).

Settings

The final table is the settings table, unlike the other tables, this table acts as an input for the game. The client can change variables in the table which directly affect users who log in afterwards. This is crucial to the usefulness of this game as a research tool. If the client wants to know how people's behavior changes if they get more, or less budget, they can simply change the values in this table without having to change anything in the code of the game.

Realization

The next step was to start developing the game. Unity was chosen as the programming environment to build the game in because of its focus on game development and for its multi-platform compatibility. Within unity, games can be programmed using C# or using UnityScript, which is similar to JavaScript. While C# is more verbose, it is less prone to ambiguity errors than UnityScript, hence C# was chosen for this project.

The game can be played at <http://christiaanverloop.000webhostapp.com/>.

Game

Next will be a description of all the scenes of the game and the functionalities within them.

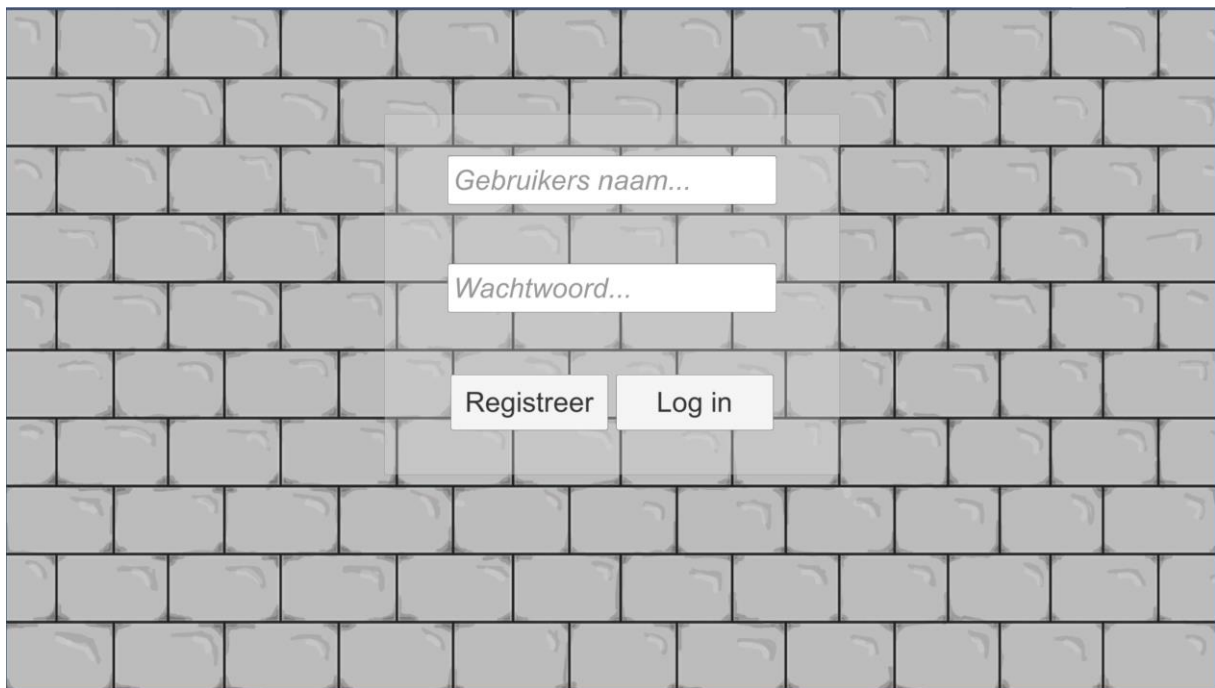


Figure 3: Login screen

This is the login screen. In here the player can register a new account or log in to an existing account. Users can register without a password if they want. It is not possible to register a new account with an existing username.



Figure 4: Introduction screen

This is the introduction screen. This explains the setting of the game to the player. When the player is ready, they can start the game by clicking 'Start'. This screen is only shown to first time players.

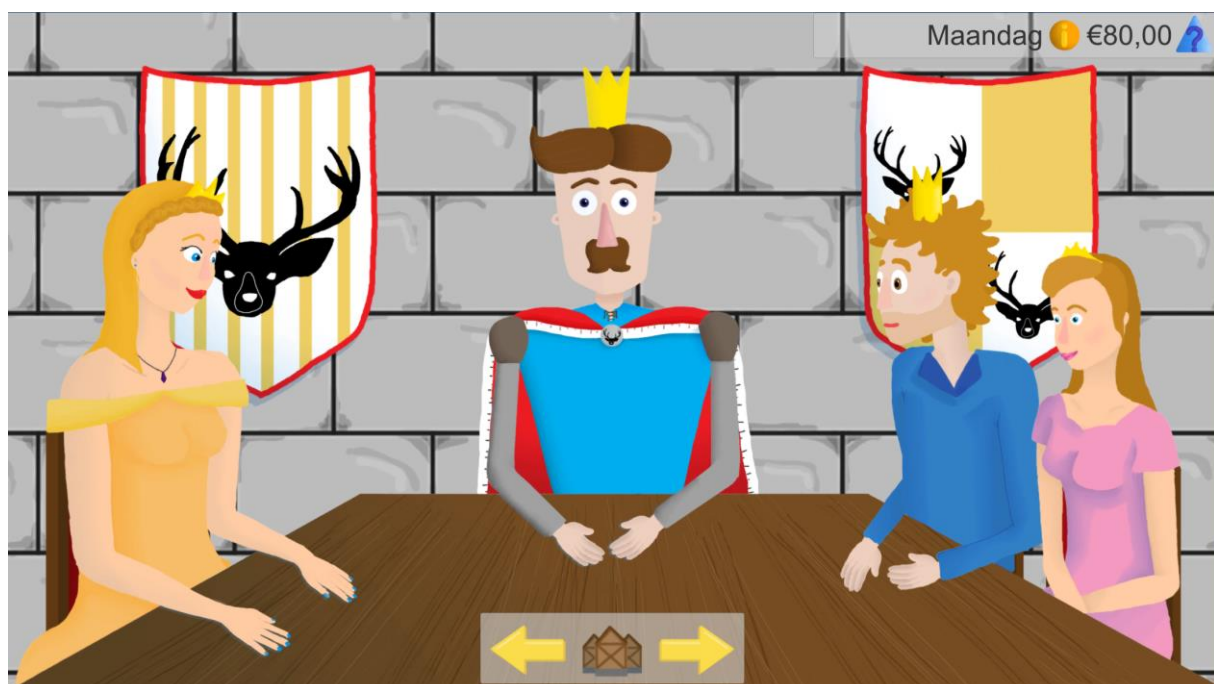


Figure 5: Dining room

After the player has clicked the start button, the player enters the dining room of the king. Here, the king will tell the player that there is no time limit today, and that the storages in the kitchen are still empty. The player has to buy food and make food today. In this scene the player can click on arrow left to go to the katapult (figure 7), the village button (figure 11), and arrow right to go to the kitchen (figure 6). In the top right, the player can see which day it is today, and how much money the player currently has. The orange infobutton shows the statistics of the game so far (figure 9). The blue button with the questionmark pops up a help screen (figure 10).

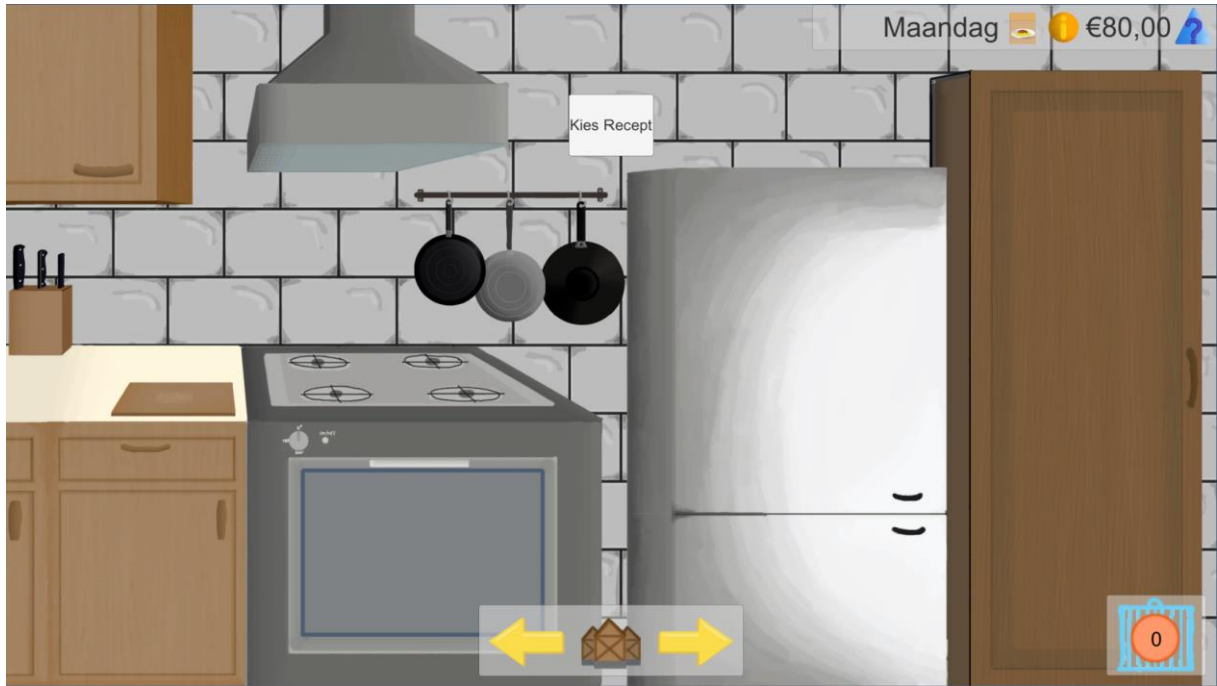


Figure 6: Kitchen

This is the kitchen, where players can store products and make food. In this scene the player can click on arrow left to go to the dining room (figure 5), the village button (figure 11), and arrow right to go to the catapult (figure 7). On the top right there is an extra button available compared to the previous scene; the recipebook. Clicking this opens a recipebook (figure 8), which shows the required ingredients for each recipe. The storages (fridge, cupboard, freezer) can be clicked to see their content (figure 27). The bag icon in the bottom right corner can be clicked to view its content (figure 26).

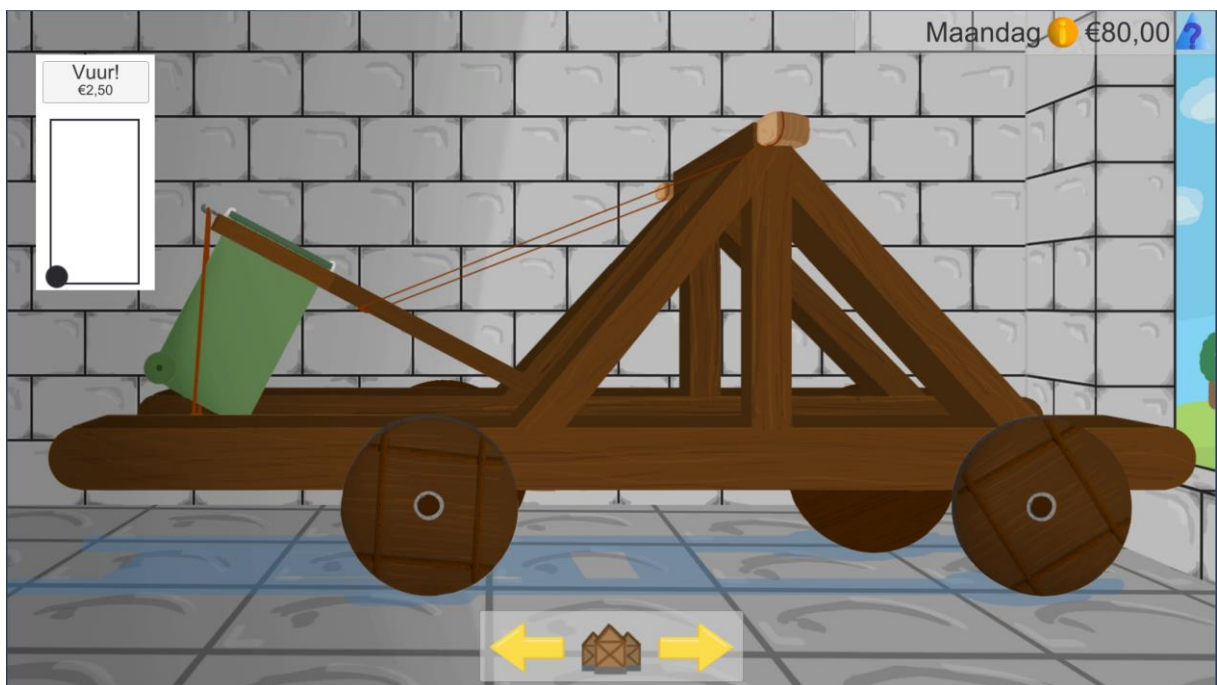


Figure 7: Catapult

The catapult stores all the foodwaste. The bar on the top left fills up as more food is put into the catapult. The catapult costs money to fire, indicated in the fire button. The player can click on arrow left to go to the kitchen (figure 6), the village button (figure 11), and arrow right to go to the dining room (figure 5). When the fire button is clicked, food is catapulted into the village (figure 36).



Figure 8: Recipe book

When the recipe button is clicked, this window pops up. It shows what ingredients are required or possible to use in all the recipes. The recipes are suitable for 4 persons. This means that the player has to adjust the amounts to match the number of eaters that day. The folded corner of the page is clickable and takes the player to the next page of the book. The arrow on the top left closes the recipe book.

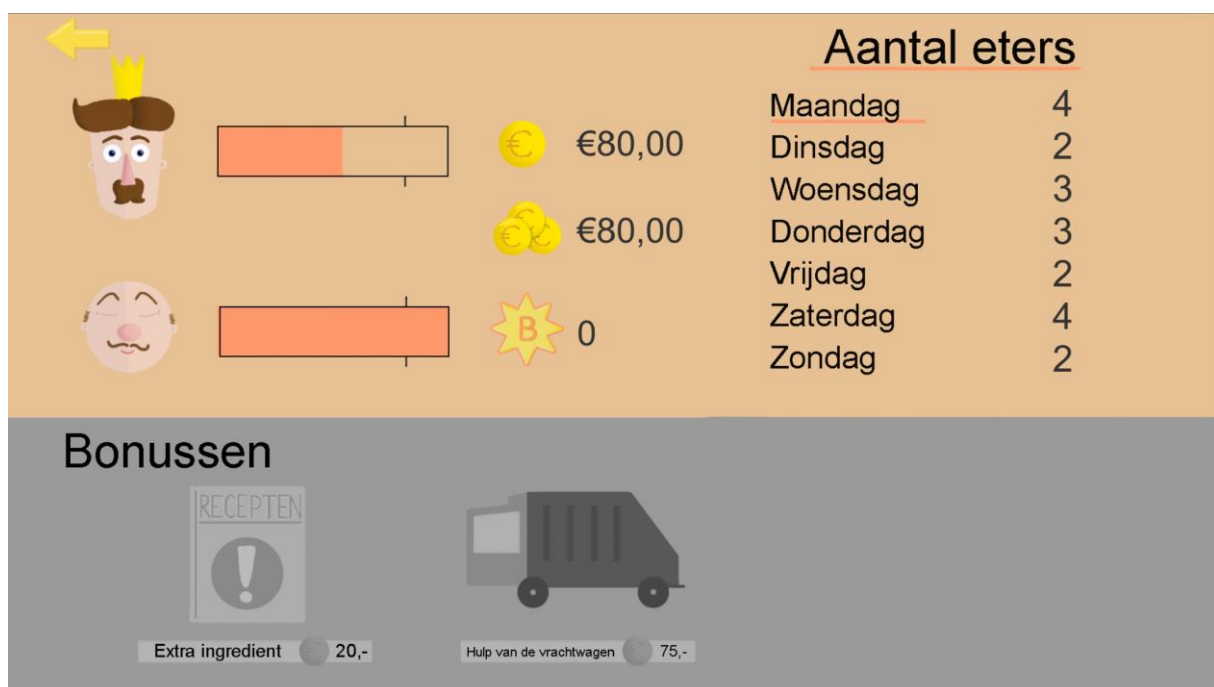


Figure 9: Statistics overview

When the information button is clicked, this window pops up. In here, the player can check how many eaters there are on each day of the week, as well as the satisfaction of the king and the village. In the middle, from top to bottom are the player's balance, the weekly budget, and the number of bonuspoints. These bonuspoints can currently not be used yet, but are meant to buy bonuses from. The arrow on the top left closes the window.

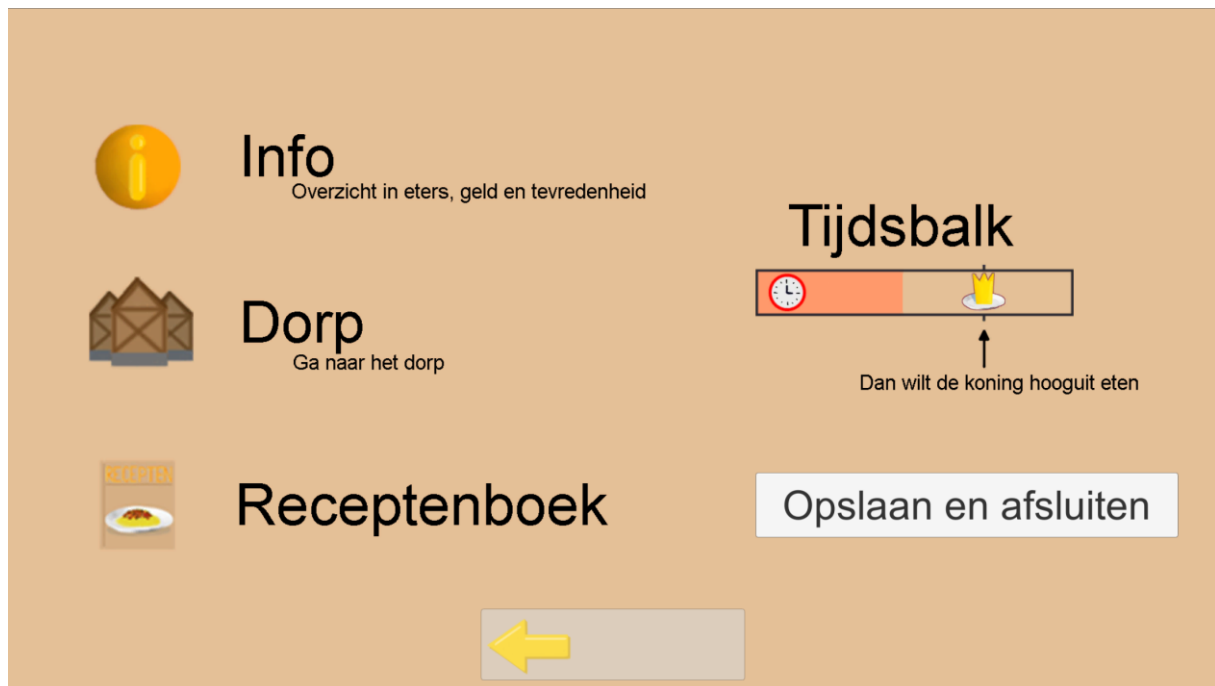


Figure 10: Help screen

The questionmark button opens this window, which shows a short explanation of the icons. The arrow closes the window again. By clicking on the 'Opslaan en afsluiten' button, the player can save the game, to later continue where they left of.

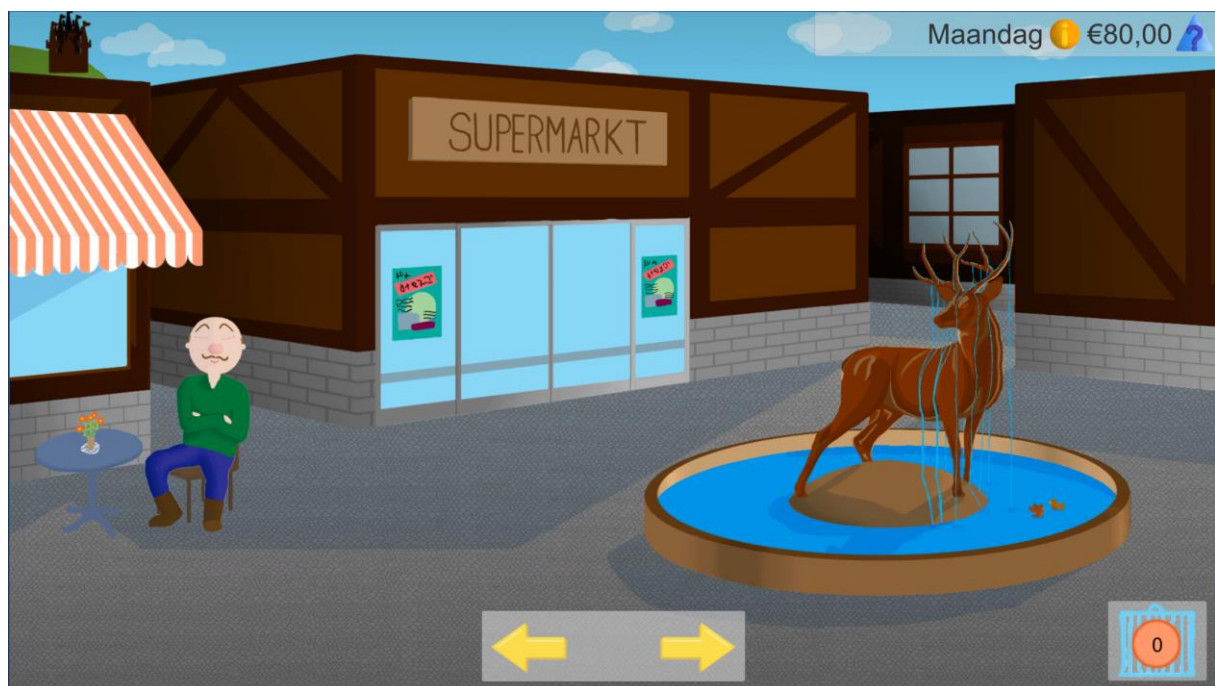


Figure 11: Village

The village button leads to this scene. In here is a square with a statue, a villager, and a supermarket. The player can click on arrow left to go to the street with the greengrocer (figure 19) and arrow right to go to the street with the butcher (figure 22). To go back to the castle, the player can click on the castle on the top left (figure 25). In the bottom right is the shopping bag of the player, which can be clicked to view the contents (figure 18). In the middle of the scene is a supermarket, which when clicked on takes the player to the supermarket. (figure 12).

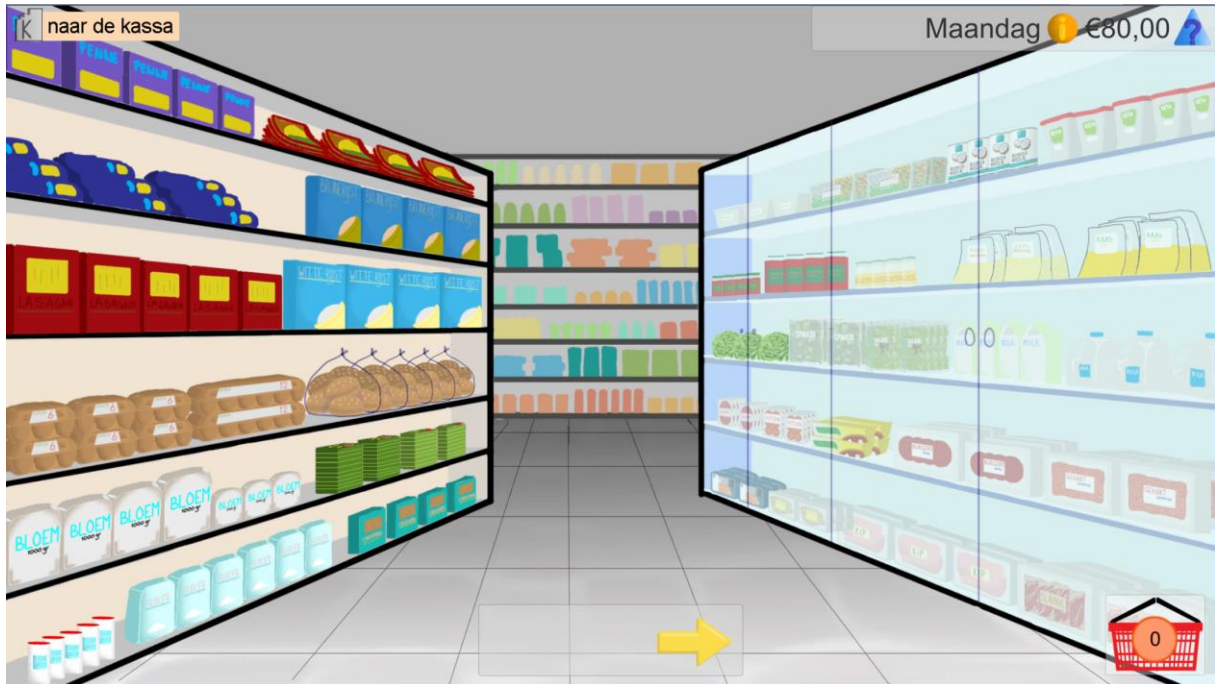


Figure 12: Supermarket lane 1

This is one of two shop lanes. On the left is a dry shelf, and on the right is a cooled shelf. The player can click on the shelf quadrants to view the products on that quadrant (figure 14). By clicking on the button on the top left, the player is taken to the cash register (figure 17). The arrow to the right leads to the second shop lane (figure 13). The basket icon in the bottom right can be clicked to show the contents of the basket (figure 16).

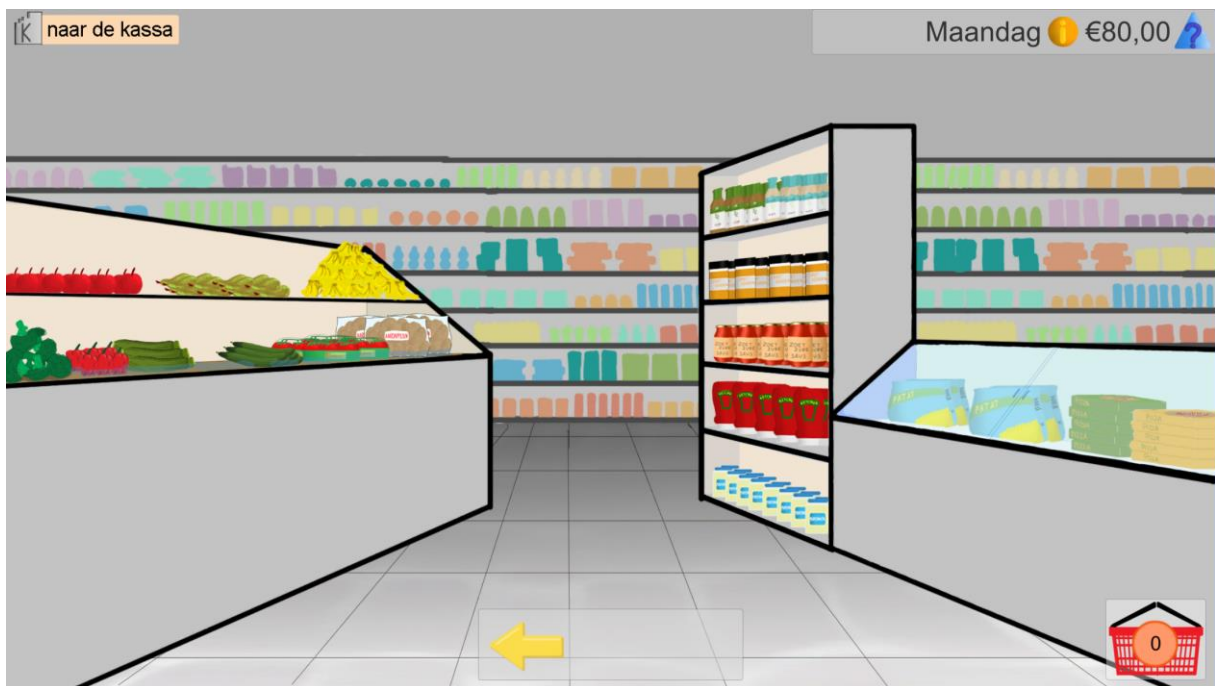


Figure 13: Supermarket lane 2

This is the second shop lane, which functions nearly identically to the first shop lane. In contrast to the first shop lane, this lane has a frozen section on the right.



Figure 14: Inside of a shelf

When the player clicks on a shop section, the products can be viewed from closerby. The player can either buy the product directly from here, or view more information about the product by clicking on it (figure 15). The arrow on the top left takes the player back to the appropriate shop lane.

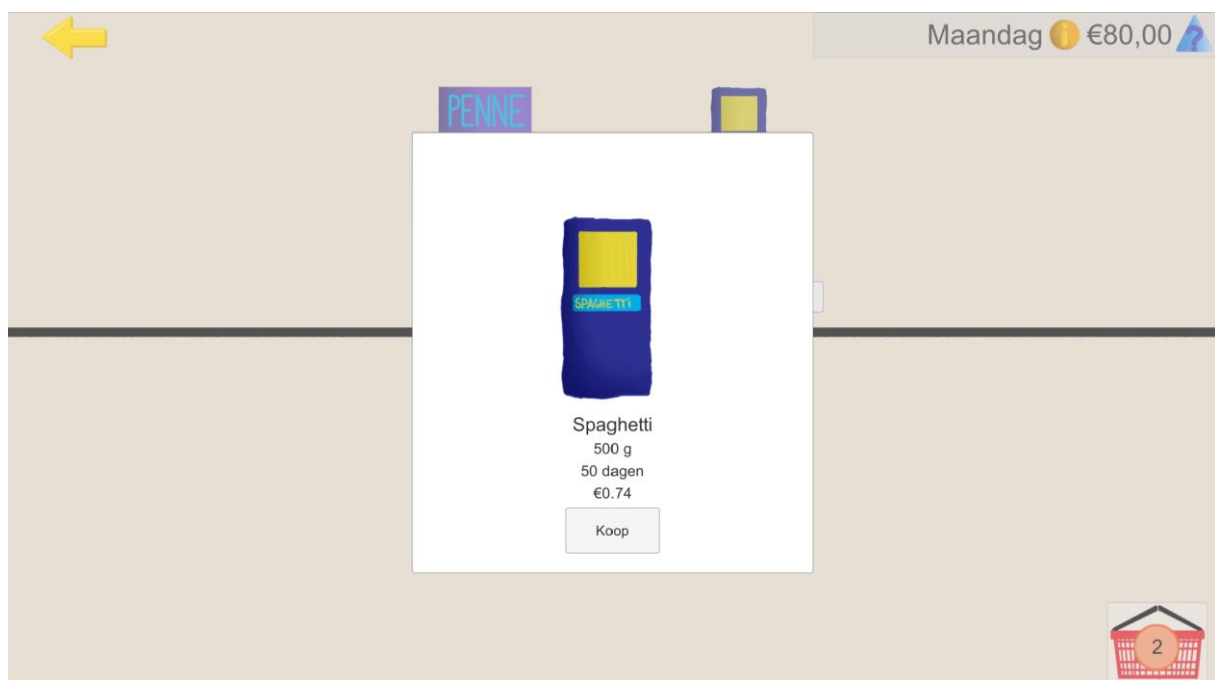


Figure 15: More product information

If a player clicks on the product picture, this popup shows more information about the product. This includes the amount, the due date, and the price of the product. This window can be closed again by clicking on the side of the popup.

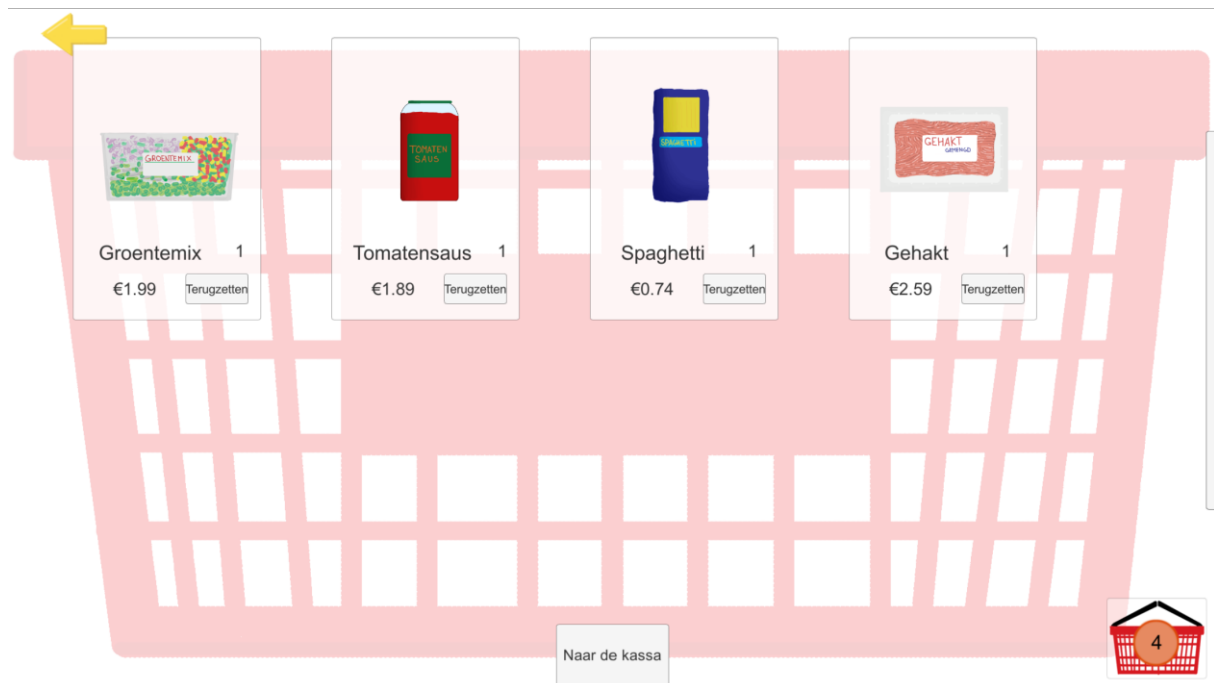


Figure 16: Basket

This popup shows the content of the basket. Products can be clicked to view more information (figure 15). Products can also be removed from the basket by clicking the 'terugzetten' button. This popup can be closed again by clicking on the basket icon in the bottom right. The player can also go to the cash register from here by clicking the button on the center bottom (figure 17).



Figure 17: Supermarket cash register

The text in the center of the screen tells the player how much the products in the basket cost in total. The player can either confirm the purchase by pressing the left button, which takes the player back to the village (figure 11) or continuing shopping by clicking the right button, which takes the player to the previous shop lane.

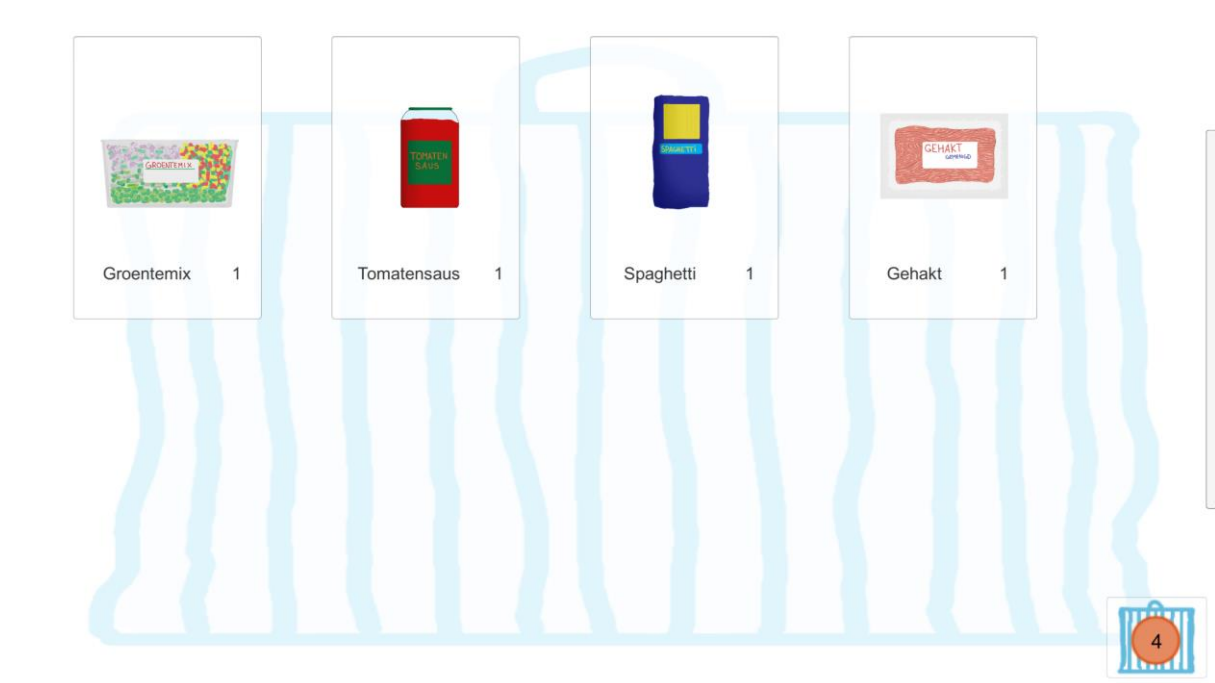


Figure 18: Shopping bag

When the player pays for the products, they are transferred to the shopping bag. In here, the player can view the products. This window can be closed again by clicking on the shopping bag icon.

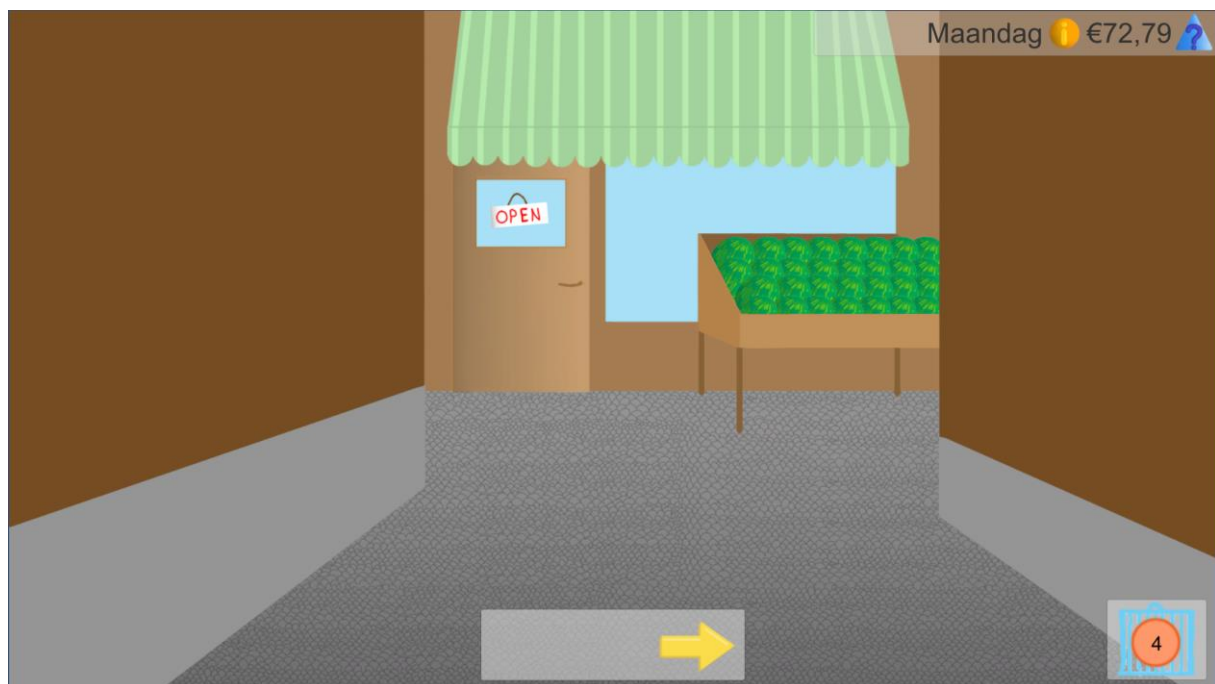


Figure 19: Street with greengrocer

On the left side of the village square is this street which contains a greengrocer (figure 20). The player can go back to the village square by clicking the arrow.

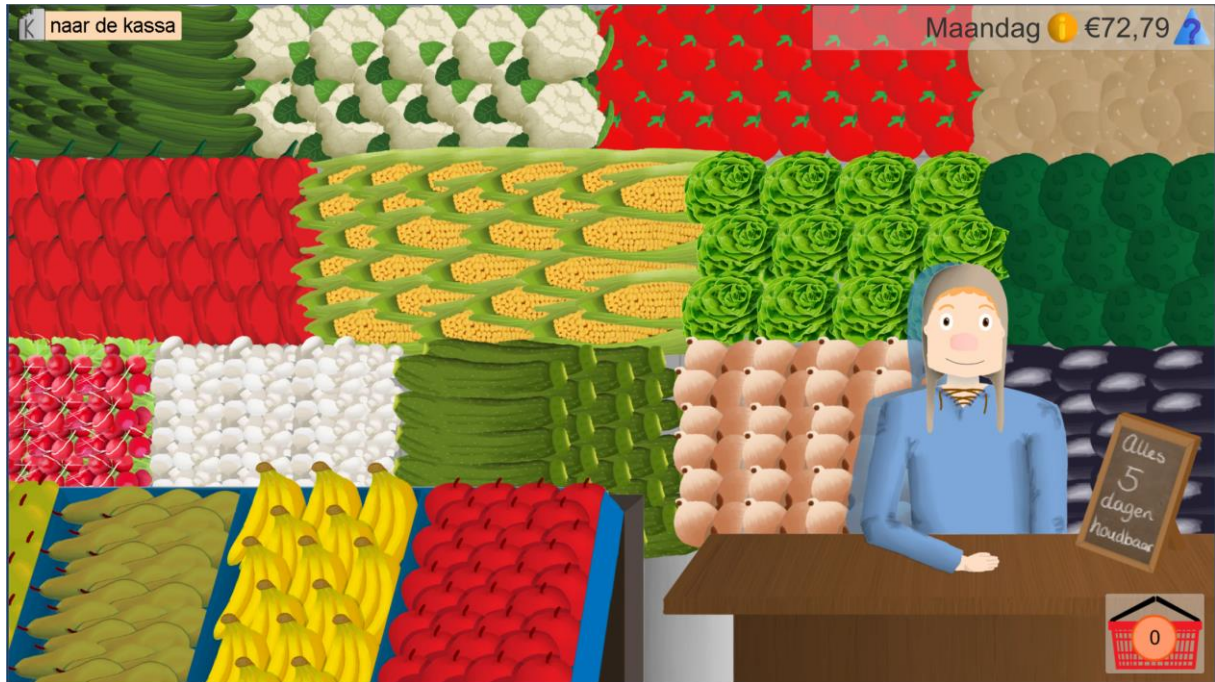


Figure 20: Greengrocer

In the greengrocer, the player can click on products to view more information about them (figure 15). The player can pay for the products by clicking on the employee or by clicking on the button on the top left. This leads to the checkout window (figure 21).

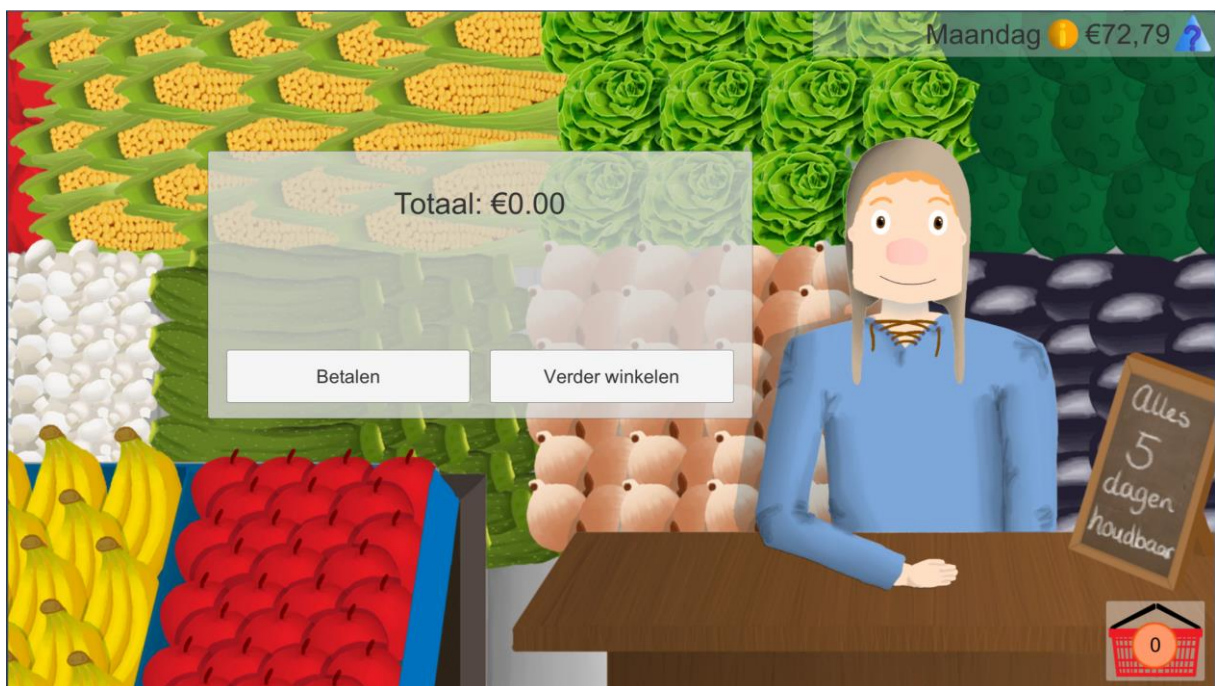


Figure 21: Greengrocer cash register

In this scene, the player can pay for the products bought in the greengrocer. This functions the same as the cash register in the supermarket (figure 17).



Figure 22: Street with butcher

On the right side of the village square is this street which contains a butcher (figure 23). The player can go back to the village square by clicking the arrow.

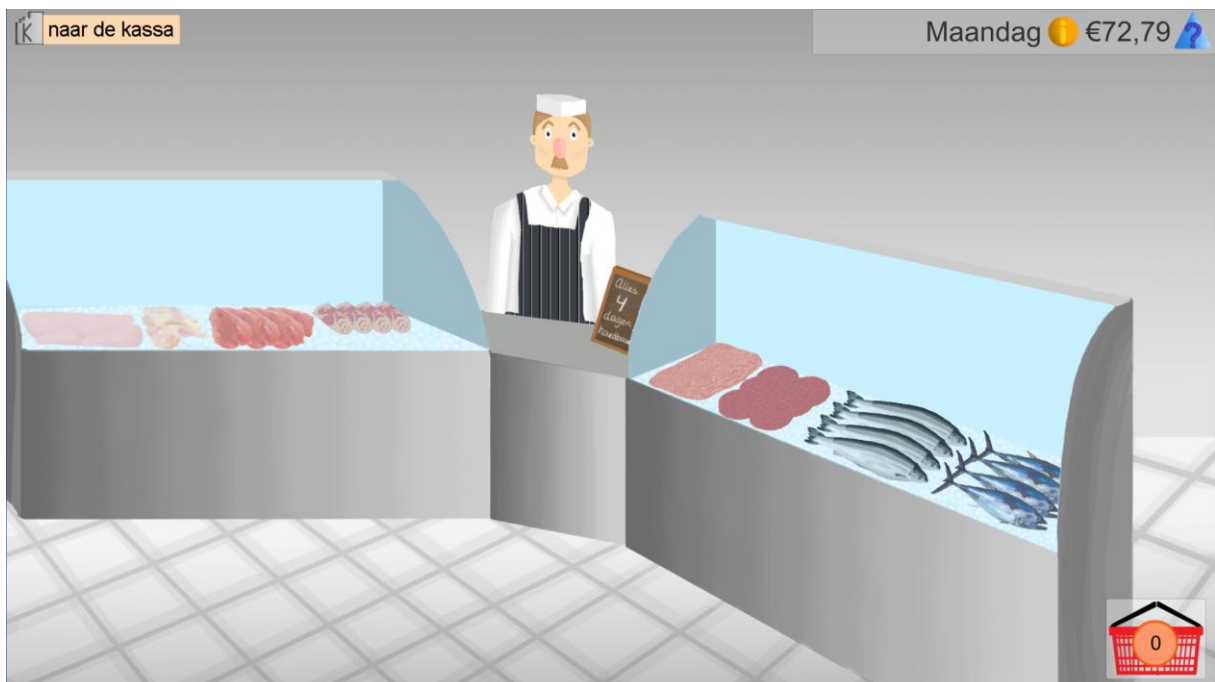


Figure 23: Butcher

In the butcher, the player can click on products to view more information about them (figure 15). The player can pay for the products by clicking on the employee or by clicking on the button on the top left. This leads to the checkout window (figure 24).



Figure 24: Butcher cash register

In this scene, the player can pay for the products bought in the butcher. This functions the same as the cash register in the supermarket (figure 17).

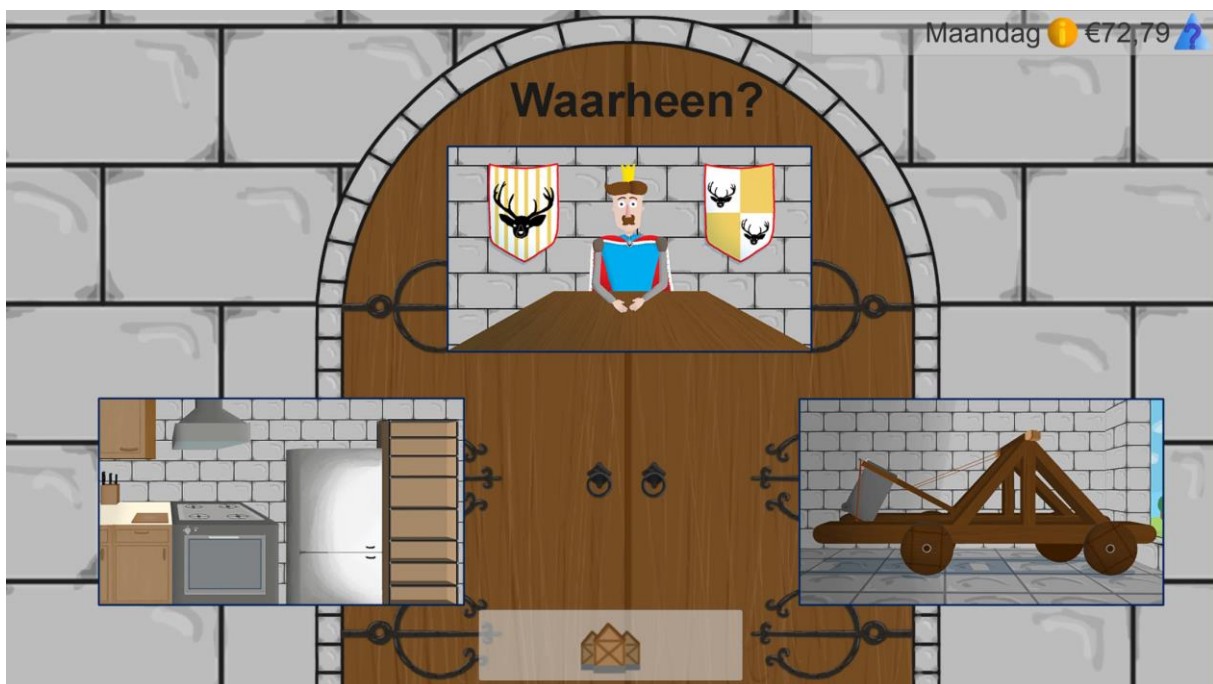


Figure 25: Castle entrance

In the village, the player can click on the castle to get to this scene. Here, the player can click on any of the scenes to go to the respective room in the castle.



Figure 26: Shopping bag in kitchen

After the player has bought products in the village, the products can be dragged from the bag to one of the storages; fridge, cupboard or freezer. The products expire faster in warmer storages and slower in colder storages.

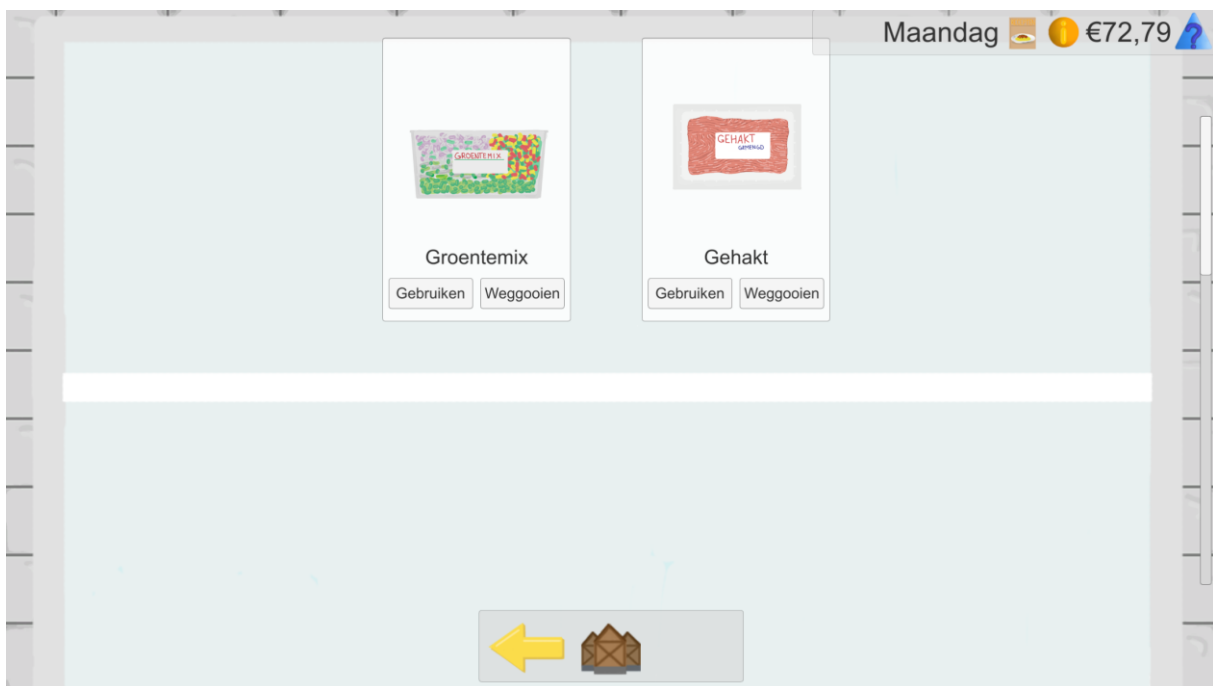


Figure 27: Products in fridge

In the storages, the player can view the content of the storage. Like in the supermarket, the product picture can be clicked to view more information about the product (figure 29). The player can decide to use a product by clicking the 'gebruiken' button (figure 30), or waste a product by clicking the 'weggooien' button (figure 35).

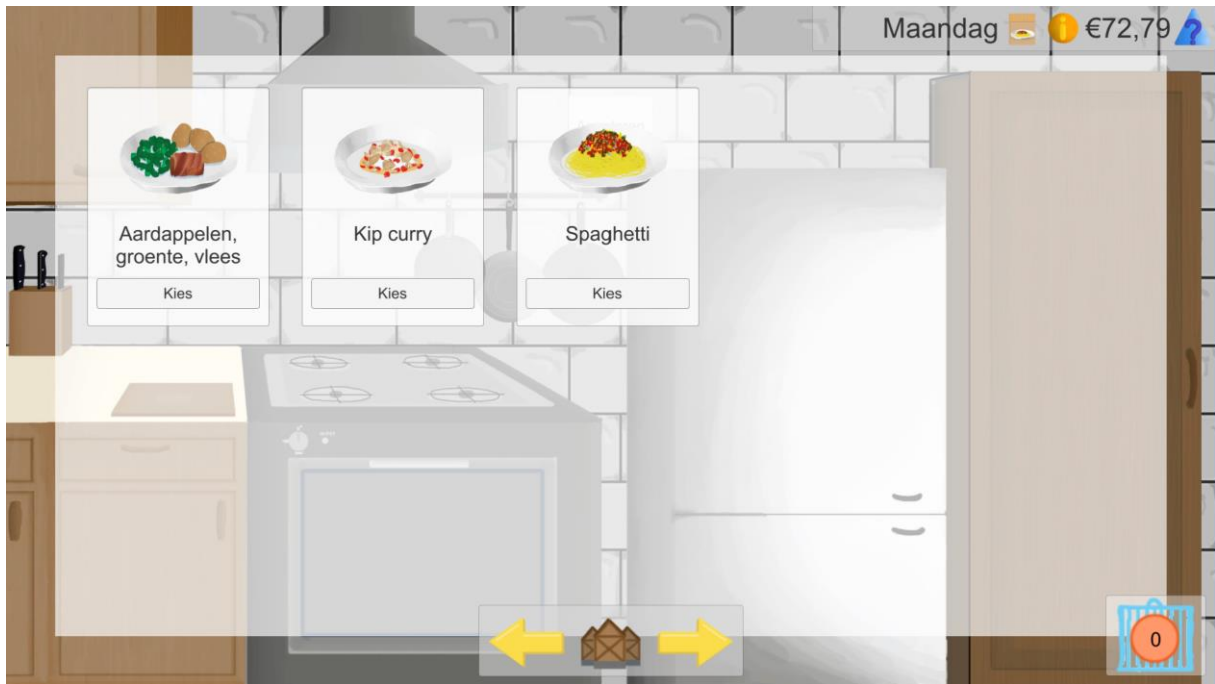


Figure 28: Recipe choice menu

After clicking the 'Kies recept' button in the kitchen, this window pops up. The player can choose the recipe they want to make by clicking 'kies' on the desired recipe.



Figure 29: More product information

In the storages, more information can be seen by clicking on the product, this is very useful for the player to keep an eye of the amounts left and the due date remaining. If a product expires, the visuals change (figure 41).

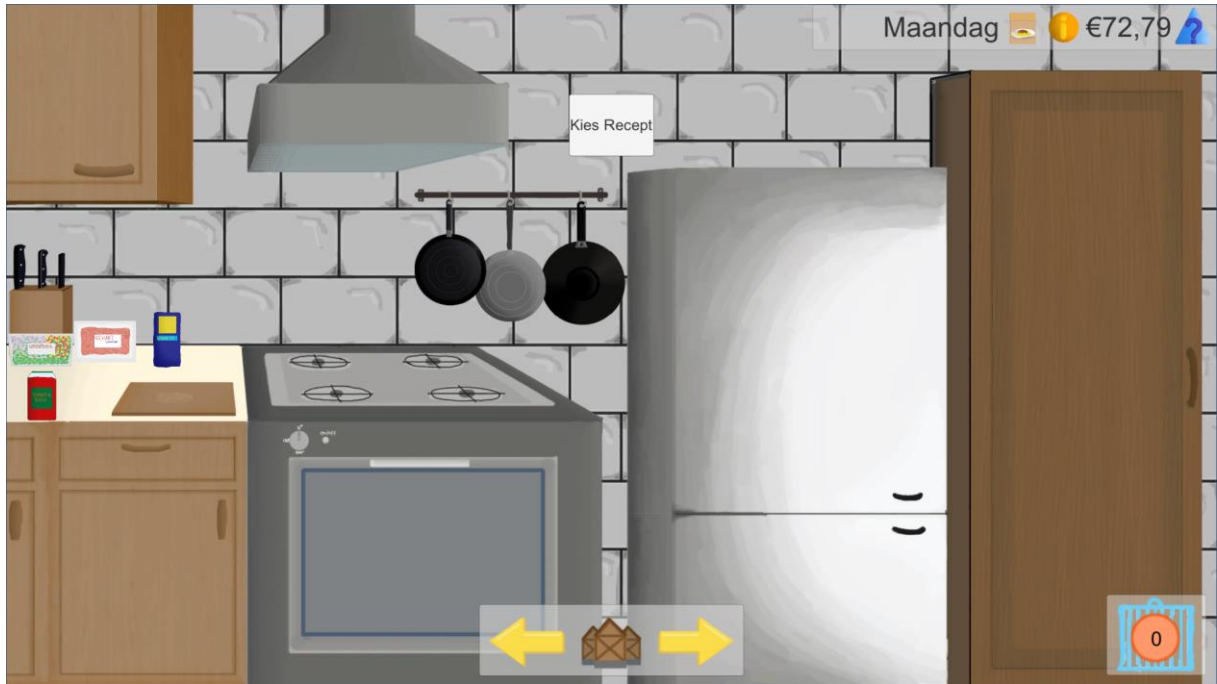


Figure 30: Products on kitchen counter

When products are used, they are put on the kitchen counter. The player can then start cooking by clicking on the cooking area (figure 31).



Figure 31: Cooking part of the kitchen

The player can click on pans to put them on the stove. The player can then drag products to the pan (figure 32). The arrow can be clicked to go back to the full kitchen view (figure 6).



Figure 32: Products in pans

When a product is dragged to a pan, a text input field pops up. The player enters how many grams they want to put into the pan. For this they can check the recipebook (figure 8) or come up with their own amount. When the first product enters a pan, a progress bar appears. A purple triangle indicates how ready the dish is. The player can stop the progress of a pan, by clicking on it. Once the player is ready to serve, the 'Serveren' button can be clicked. This will serve the meal to the king (figure 33).

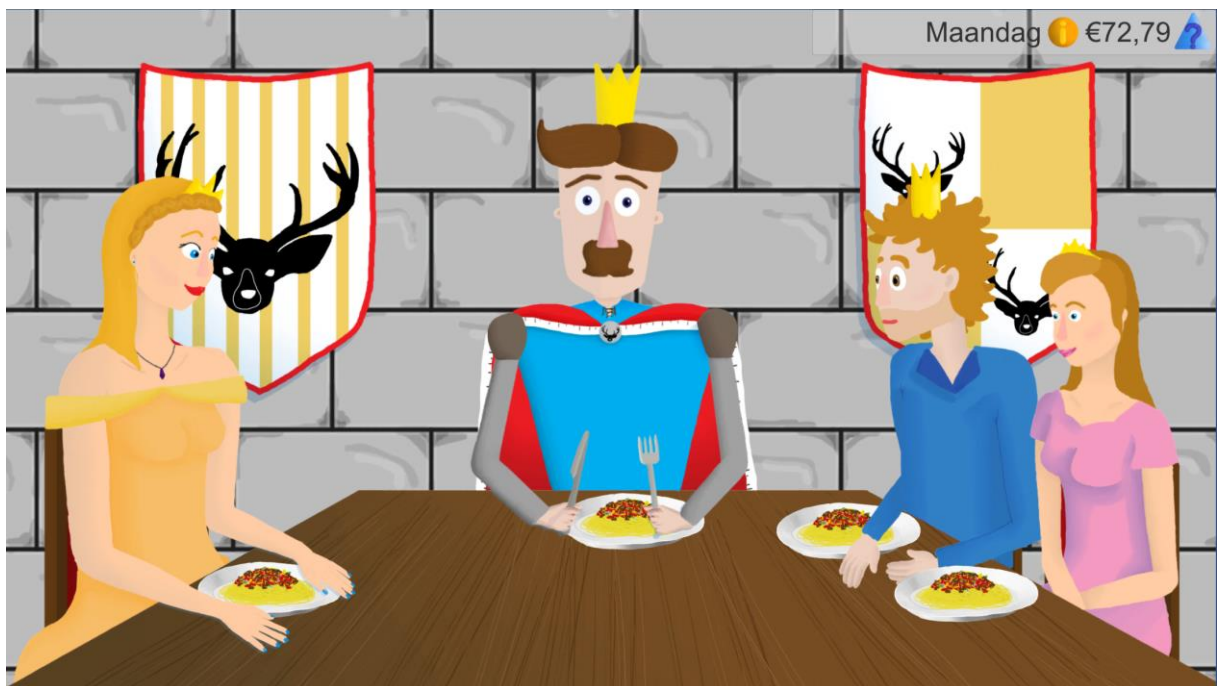


Figure 33: King and guests eating

When a meal is served, the king judges it based on 6 factors: Was the meal cooked long enough, but not too long (figure 32). Was an expired ingredient used (figure 41). Where ingredients missing. Was enough food served for the number of eaters, and was the meal served in time. Based on these factors, the eaters eat nothing, part, or everything of the meal (figure 34). The king also makes verbal comments on the meal.

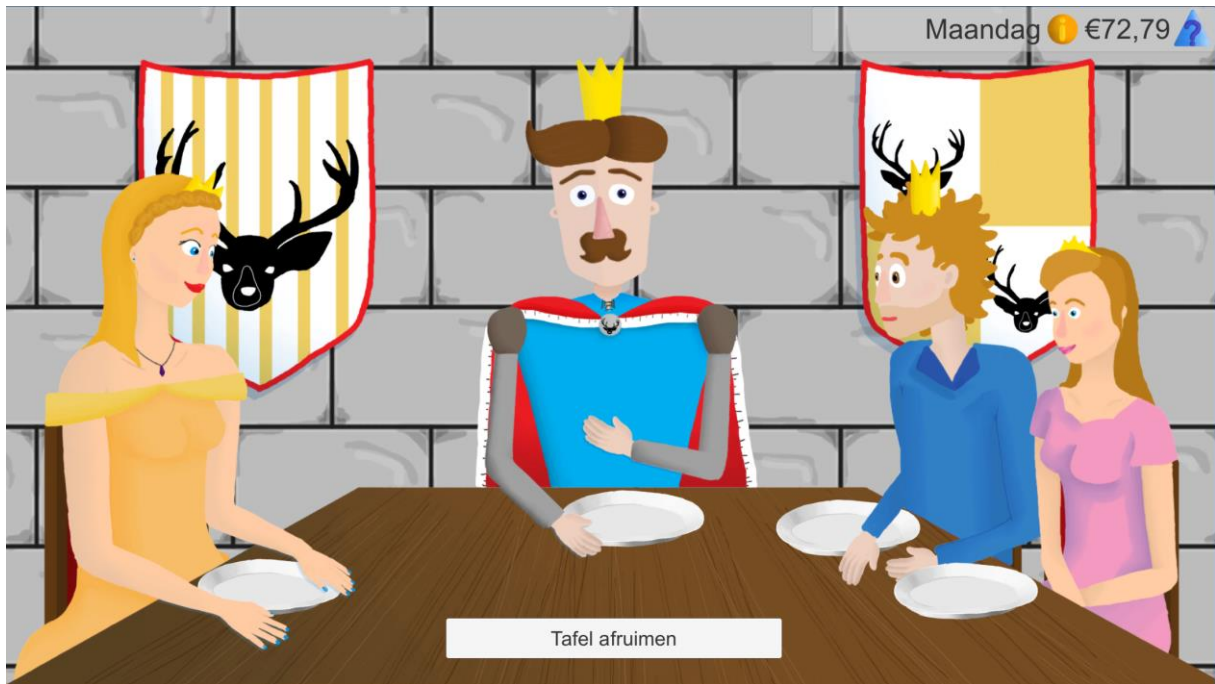


Figure 34: Finished dinner

Once the meal has been finished (or discarded as unedible), the 'Tafel afruimen' button appears. If there were no leftovers, this button will only clear the plates from the table, but if there are leftovers, these leftovers are transferred to the catapult (figure 35). When this button is clicked, the next day starts.

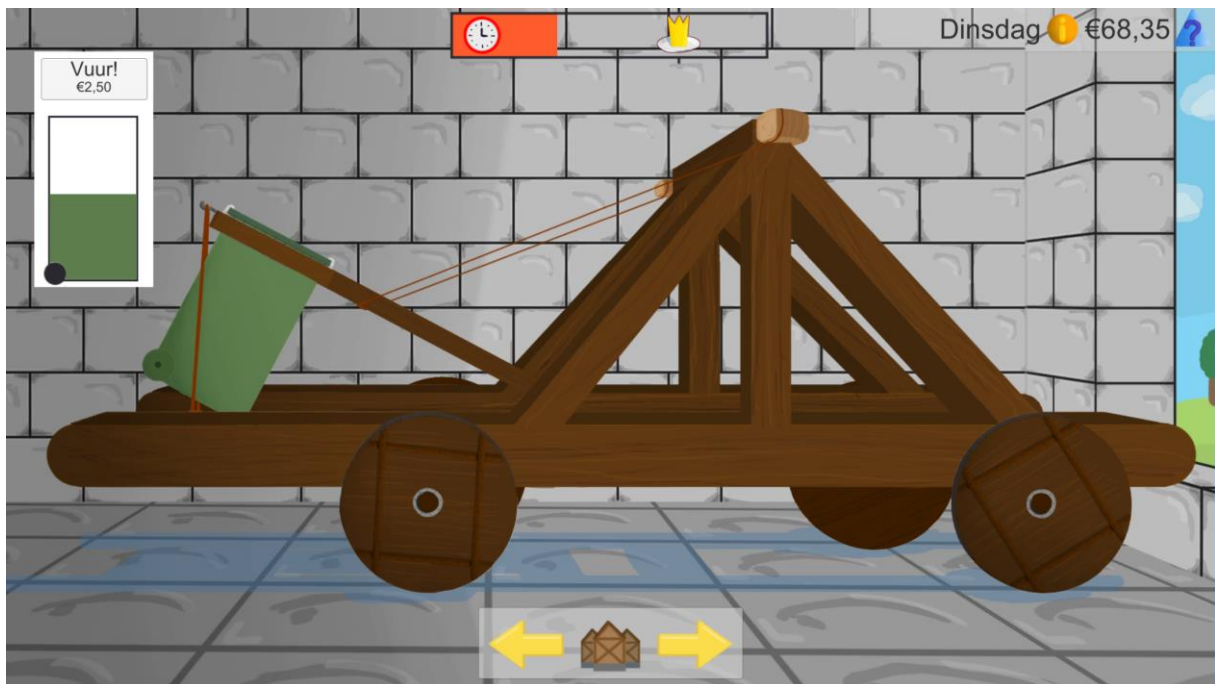


Figure 35: Food waste in the catapult

From the second day on, there is a timelimit to each day, indicated at the center top. If there is food in the catapult, it can be emptied by pressing the 'Vuur!' button (figure 36).

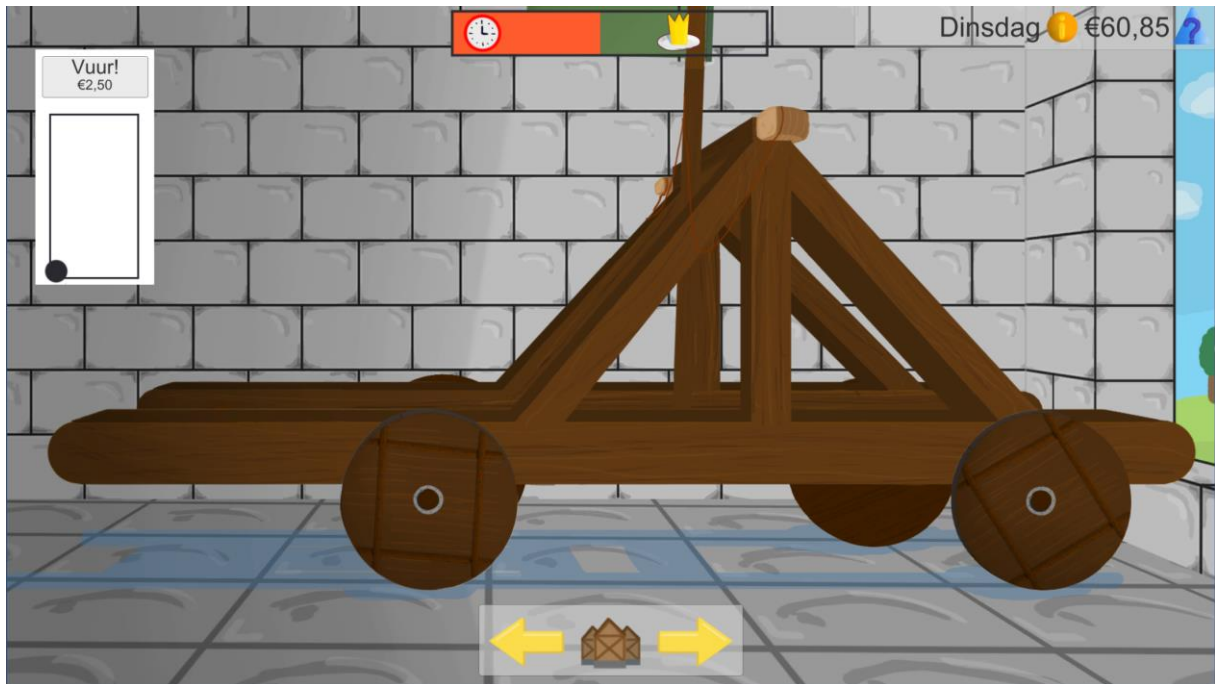


Figure 36: Catapult being emptied

When the catapult is fired, the food waste is catapulted into the village (figure 37).

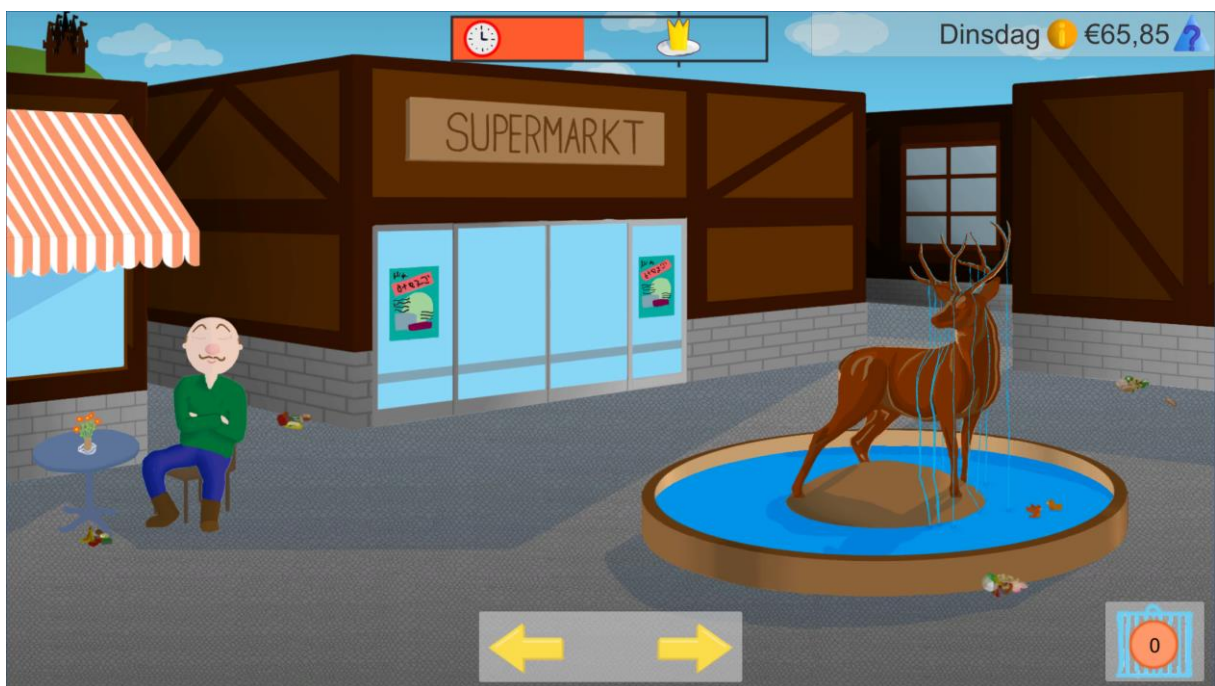


Figure 37: Food waste in village

Food waste catapulted is visible in the village. The more waste is catapulted, the more waste is visible in the village (figure 38).

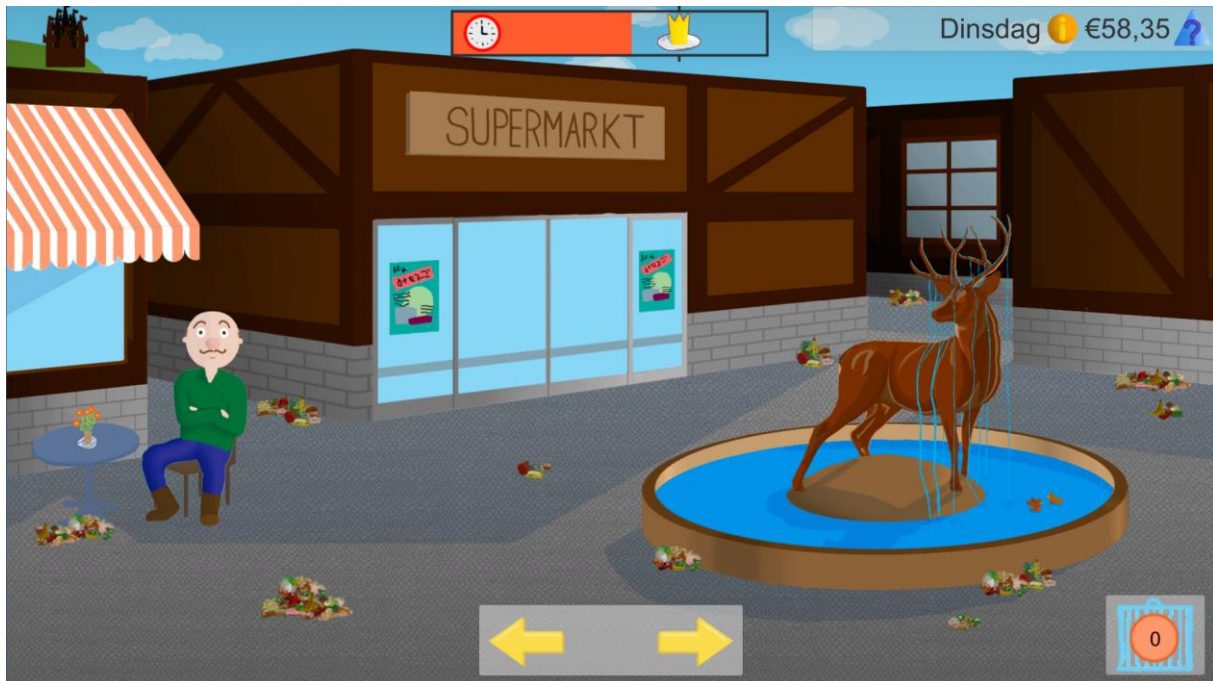


Figure 38: More food waste in village

If the village gets very dirty, the satisfaction of the villagers decreases, this can be seen in the villager's expression or in the overview (figure 39).

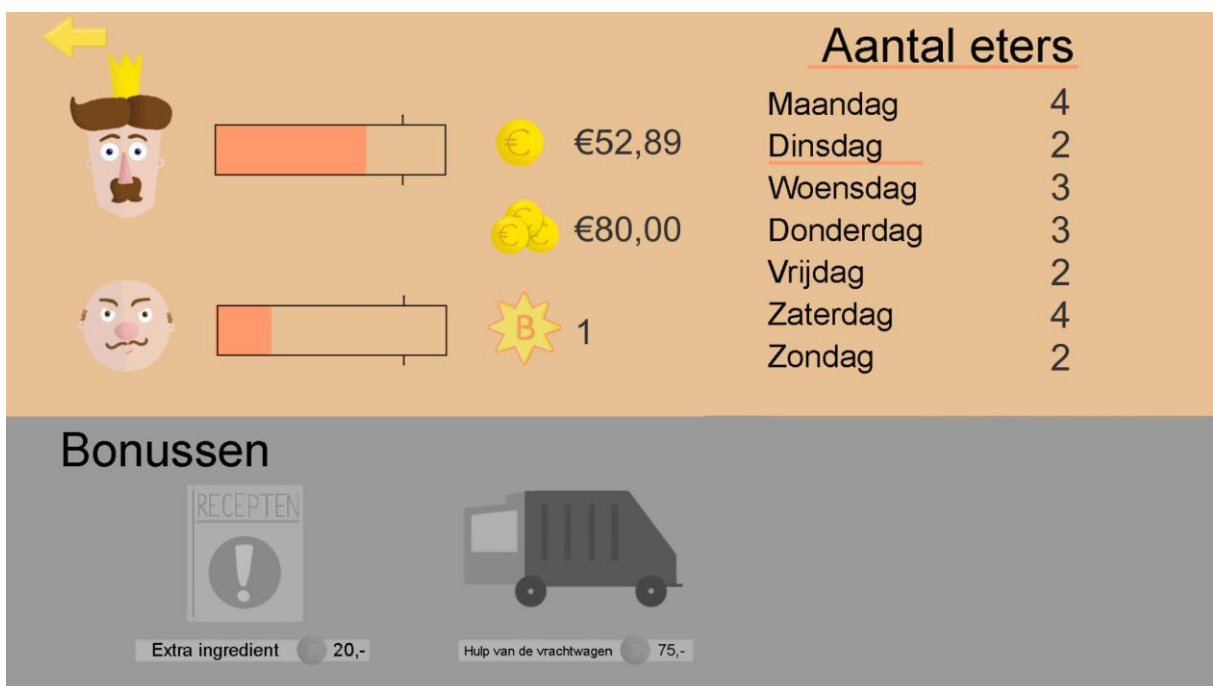


Figure 39: Decreasing satisfaction

In the overview, the villager satisfaction decreases as more food is catapulted into the village.

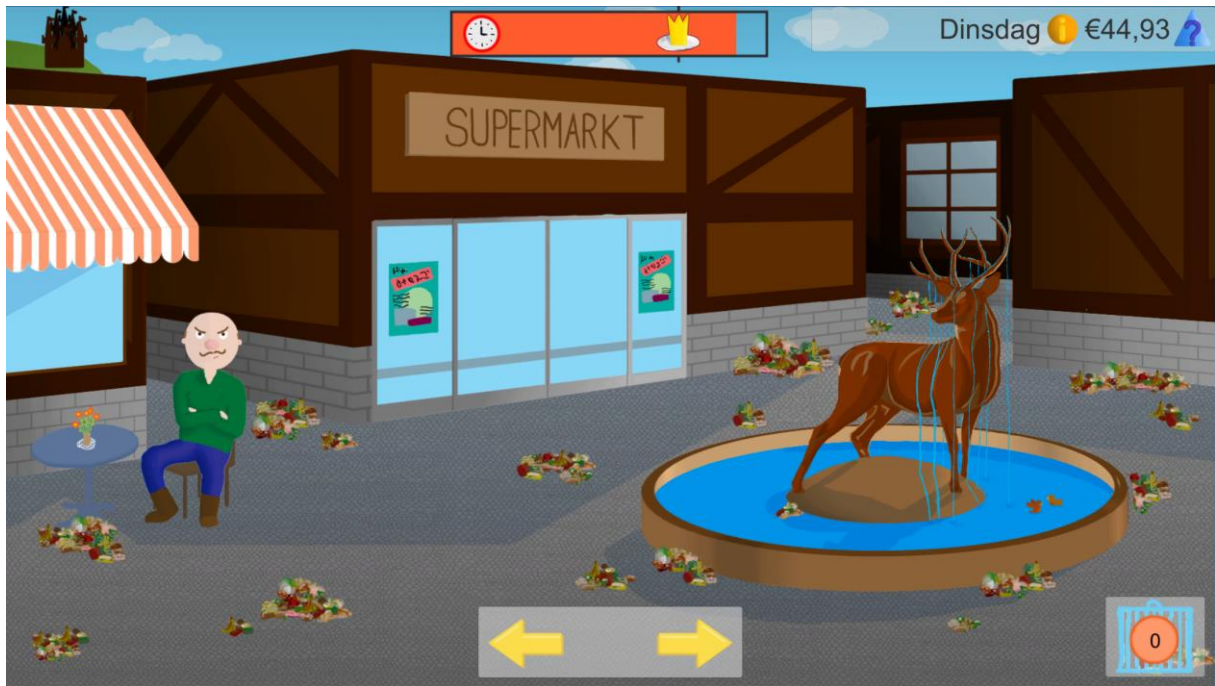


Figure 40: Most food waste in village

This is the maximum visual dirtyness of the village.



Figure 41: Expired products

If products are stored for too long, they expire, which changes their appearance. The due date can also be seen by clicking on the product picture (figure 29).

Database

The data structure in figure 2 was used to construct the database as can be seen in figures 42 through 48. The database is hosted on a server to allow multiple game clients to write data to it simultaneously. The main events are stored in the event table. This is event-based data; the actions in the game can be reconstructed from this list. One of the events is 'ending a day'. This action has a DayStatsID which points to another table; the DayTable. In this table are some state-based data is stored, like the budget of the player at the end of the day, the amount of food waste, and the king's satisfaction level.

This structure was chosen because it can be very tricky to gather this information using event-based data only; one would have to calculate every change in the player's balance from a point at which it was known in order to find out how much money the player currently has.

Some questions are still difficult to answer with the current table structure. Like 'what products are in the storages right now'. This question requires you to trace every product that moved in and out of a storage. However, the client is more interested in overall trends rather than specific snapshots.

The combination of event-based data and a relational database does cause some columns to be empty, since not all the event properties apply to every event, however, as long as the number of columns is not very large, this is not a problem. If the game is developed further, and many more types of events are added, it could be considered to switch to a non-relational database to avoid the large number of empty columns.

The raw data can be viewed using the phpMyAdmin GUI on the server. The tables can be exported to a variety of formats including CSV. Specific queries can be retrieved using MySQL, some examples of which can be seen in the next chapter. These query results can also be exported to CSV.


#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1 UserID	int(10)			No	None		
<input type="checkbox"/>	2 EventID 	int(16)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	3 EventType	varchar(50)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	4 DayStatsID	int(16)			No	None		
<input type="checkbox"/>	5 ProductInstanceID	int(16)			No	None		
<input type="checkbox"/>	6 ActionAmount	decimal(8,4)			No	None		
<input type="checkbox"/>	7 ActionDue	int(6)			No	None		
<input type="checkbox"/>	8 LocationFrom	varchar(50)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	9 LocationTo	varchar(50)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	10 Shop	varchar(20)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	11 Day	int(10)			No	None		
<input type="checkbox"/>	12 Time	int(16)			No	None		

Figure 42: EventTable


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	DayStatsID 	int(16)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	Eaters	int(2)			No	None		
<input type="checkbox"/>	3	GramsServed	int(6)			No	None		
<input type="checkbox"/>	4	Raw	tinyint(1)			No	None		
<input type="checkbox"/>	5	Burnt	tinyint(1)			No	None		
<input type="checkbox"/>	6	MissingProduct	tinyint(1)			No	None		
<input type="checkbox"/>	7	OnTime	tinyint(1)			No	None		
<input type="checkbox"/>	8	Expired	tinyint(1)			No	None		
<input type="checkbox"/>	9	KingSatisfaction	float			No	None		
<input type="checkbox"/>	10	VillageSatisfaction	float			No	None		
<input type="checkbox"/>	11	MoneyLeft	float			No	None		
<input type="checkbox"/>	12	FoodWaste	int(10)			No	None		

Figure 43: DayTable


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	ProductInstanceID 	int(16)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	ProductTypeID	int(16)			No	None		

Figure 44: ProductInstances


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	ProductTypeID 	int(16)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	Name	varchar(50)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	3	Grams	int(16)			No	None		
<input type="checkbox"/>	4	Price	decimal(10,4)			No	None		
<input type="checkbox"/>	5	Due	int(16)			No	None		
<input type="checkbox"/>	6	ShopName	varchar(50)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	7	ShopStorageType	varchar(50)	latin1_swedish_ci		No	None		

Figure 45: ProductTypes


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	gameFileID 	int(10)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	UserID	int(10)			No	None		
<input type="checkbox"/>	3	SaveFile	text	latin1_swedish_ci		No	None		
<input type="checkbox"/>	4	Date	int(11)			No	None		
<input type="checkbox"/>	5	Time	int(11)			No	None		

Figure 46: Savefiles


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	UserID 	int(10)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	UserName	varchar(32)	latin1_swedish_ci		No	None		
<input type="checkbox"/>	3	Password	varchar(32)	latin1_swedish_ci		No	None		

Figure 47: Users


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1	SettingID 	int(6)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	CatapultCapacity	int(6)			No	None		
<input type="checkbox"/>	3	CatapultCost	float			No	None		
<input type="checkbox"/>	4	CupboardCapacity	int(6)			No	None		
<input type="checkbox"/>	5	BasketCapacity	int(6)			No	None		
<input type="checkbox"/>	6	BagCapacity	int(6)			No	None		
<input type="checkbox"/>	7	FreezerCapacity	int(6)			No	None		
<input type="checkbox"/>	8	FridgeCapacity	int(6)			No	None		
<input type="checkbox"/>	9	KitchenCapacity	int(6)			No	None		
<input type="checkbox"/>	10	Budget	float			No	None		
<input type="checkbox"/>	11	DefaultKingSatisfaction	float			No	None		
<input type="checkbox"/>	12	VillageWasteScaleFactor	int(6)			No	None		
<input type="checkbox"/>	13	DayDuration	int(6)			No	None		
<input type="checkbox"/>	14	RawPenalty	float			No	None		
<input type="checkbox"/>	15	BurntPenalty	float			No	None		
<input type="checkbox"/>	16	LatePenalty	float			No	None		
<input type="checkbox"/>	17	ExpiredPenalty	float			No	None		
<input type="checkbox"/>	18	InsufficientPenalty	float			No	None		
<input type="checkbox"/>	19	MissingProductPenalty	float			No	None		
<input type="checkbox"/>	20	FreshBonus	float			No	None		
<input type="checkbox"/>	21	MinGramsPerEater	int(6)			No	None		
<input type="checkbox"/>	22	MaxGramsPerEater	int(6)			No	None		
<input type="checkbox"/>	23	WeeklyEaters	int(10)			No	None		

Figure 48: Settings

Some things cannot be changed using the SettingsTable; changing the properties of products or recipes need to be done within unity. The instructions on how to do this can be found in the user guide in appendix D.

Documentation

In order to make the end product useable for the client it is important that there are instructions on how to use the tool. For this reason a user guide has been made. This can be found in appendix D. The user guide described how to set up the game on a server, what instructions to give to the user, how to retrieve information from the server as well as how to change settings using the SettingsTable. It also describes how to make changes within unity, and how to reupload the game to the server. Finally, it contains a list of tested compatibilities.

If the client wants to continue development on this game, a more technical documentation is required to explain the structure of the game code, as well as to outline things that may need to be changed. This way, future programmers can build upon the current game. This technical documentation can be found in appendix E.

Evaluation

In order to evaluate the game, two user tests were conducted. One to discover any bugs or unclarities, and one to see if the game collects realistic data, as well as to see if the unclarities have been resolved after the improvements based on the first test.

User test 1

User test 1 was held when the greengrocer and the butcher didn't exist yet. There were still only two recipes. The game was still played locally, and the data was also saved locally. The king's voice lines were very basic, and there were no visible guests. There was a time limit even on the first day.

Four students were asked to play the game for about 20 minutes while thinking out loud. During the gameplay, notes were taken about unclarities, comments, and noteworthy behavior. After the gameplay, the participants had opportunity to give more comments about the game, and to explain what aspects weren't clear yet.

Results

It became obvious that the goal of the game wasn't immediately clear. It wasn't explained well to the player what they should do. Many buttons were unclear; for example, many people took a long time to find the village button, the overview button, and the recipe book button. In the village, many people didn't know how to get back to the castle. Furthermore, some people didn't understand what the timer bar was for, whereas in other participants, the timer bar created stress, especially in the first day. None of the participants noticed how many people they should be cooking for. Additionally, a point of frustration was the fact that pans and their content would disappear when going from the zoomed in kitchen view to the full kitchen view. People were also frustrated that they could only make two recipes. Lastly, when users entered a storage, the scrollbar would start half way. Products are filled from the top however, so people would have to scroll up in the storage to find the products, which created some confusion.

Based on these comments and observations, the following improvements were made:

- The king now has voice lines that guide the player through the first day.
- The time limit isn't present in the first day.
- The timer was visually changed to be more recognizable as a timer bar.
- More recipes, and variations within recipes were added to allow more freedom in cooking
- The icon of the overview button was changed.
- A help button was added, which shows the meaning of the buttons and the time bar.
- Pans now save their content when leaving the scene.
- The scrollbar in the storages starts at the top.
- There are guests in the dining room indicative of how many people the player should cook for.

Furthermore, the following things were changed:

- The game is played online, and data is stored on a server.
- A greengrocer and a butcher were added to increase people's options.
- A villager was added whose face indicates the satisfaction of the villagers

User test 2

After changes had been made based on the first user test, the second user test was conducted in the version described in the previous chapter. The goal of this user test is to see if changes resolved the unclarities found in the first user test, and to see if the behavior shown in the game is representative of behavior in the real world. To do this test accurately, the participants need be in the target audience this time. Hence, 4 adults were asked to play the game for about 40 minutes. Before the gameplay, they were asked to fill in a survey, see appendix F. The goal of this survey is to establish the real-life behavior of the participant, such that the behavior measured by the game could be compared against it. (Critical note: the client found surveys about food waste behavior to be unreliable, so it is possible that the results of the survey do not accurately reflect their real-life behavior, however, surveys are currently the best tool available to compare the game against.) During the gameplay, notes were written down about comments made by the participants, unclarities, and noteworthy behavior.

Results

In contrast with the first user test, the timer bar was understood by all participants. The overview icon was clicked more often, and the village button was found more quickly, however, people also clicked the village button when they wanted to go back to kitchen. Multiple participants explained that the village button looked a bit like a castle. The bag icon also caused some confusion; one participant thought it looked like a prison.

Overall, the goal of the game remained unclear in the beginning, and some buttons were still confusing. The participants expressed great frustration about the limited number of recipes, and the freedom within them. They wanted to make their meals unrestricted of any recipes. Another point of frustration was that leftovers could not be saved for the next day, but had to be thrown away. This point was particularly made by one participant who intentionally always makes too much food, and eats the leftovers for lunch the next day.

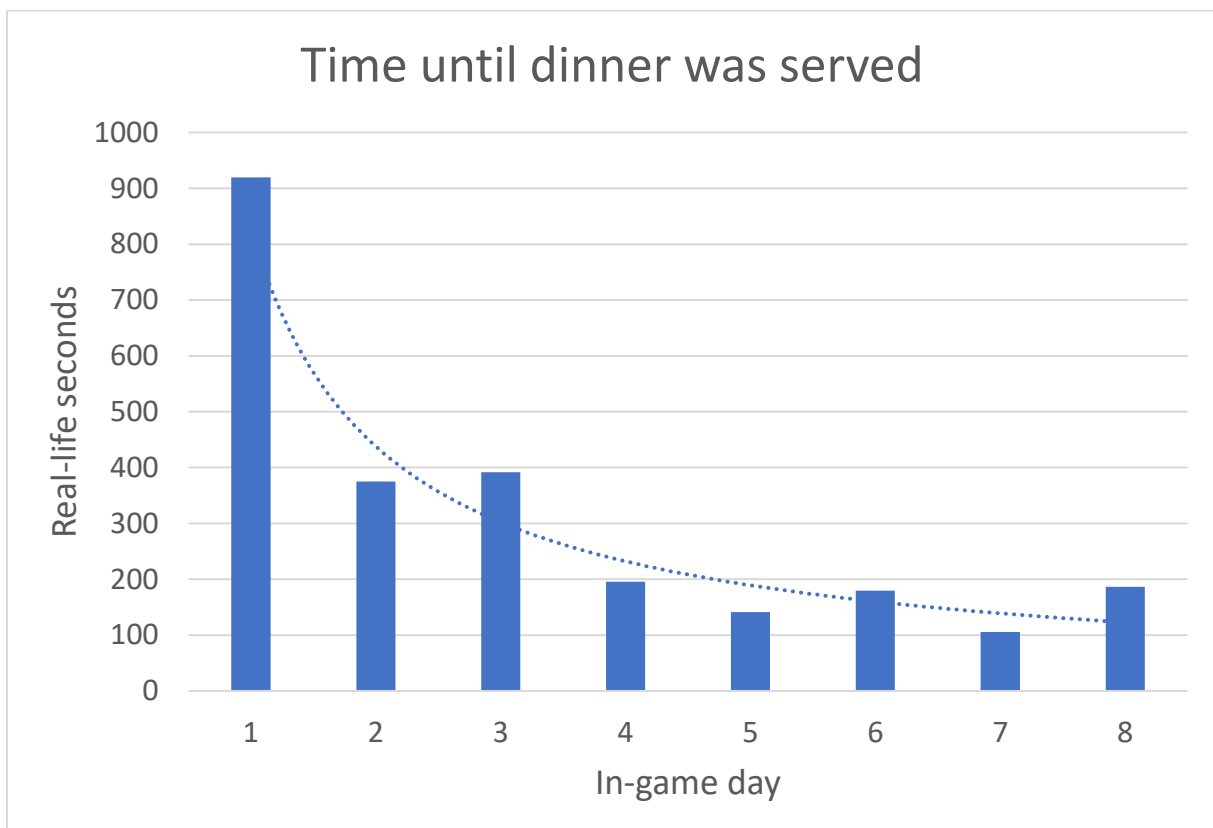
Behavior

The results from the survey were compared against the observed behaviors. It became clear that some aspects of the game evoke highly unrealistic data. One such example is the number of times people go shopping, see graph 1. In the survey all participants indicated that they go shopping between 1 and 3 times per week. However, in the game, people went shopping multiple times per day.



Graph 1: Average number of shop visits per day.

In 40% of these shop visits, nothing was bought. When asked about this behavior, the participants admitted that this behavior was not representative of their real-life shopping behavior. Some participants reported that this was caused because they had to remember the products for the recipe by heart, rather than being able to make a shopping list, or take the recipe book to the shop.

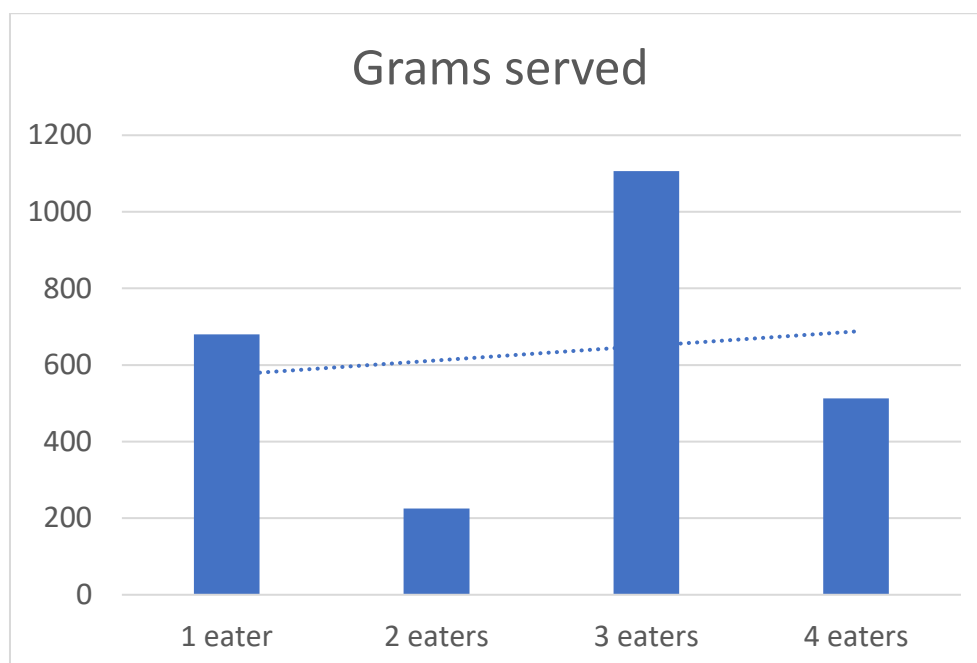


Graph 2: Time until dinner was served.

Graph 1 and Graph 2 show that the game has a very significant learning curve. In the beginning, people are displaying highly unrealistic behavior, however, after a few in-game days, the behavior does start to become more realistic.

After a few days though, the catapult starts to get full. When people fire the catapult, there is visible food waste the next time they visit the village. Some participants noticed this immediately, while others only noticed a few days later. Either way, once it is clear to the player that keeping the village clean is another goal of the game, they change their behavior. The sample size was too small to see this in the data, but all players did make verbal comments about the village getting dirty, and recognizing that this is a bad thing. The question is of course; is the changed behavior still reflective of real-life behavior once people recognize this secondary goal of the game? It could be that the act of making people aware that food waste is bad changes their real-life behavior too, however, it seems more likely that they will only change their behavior in the game, without changing their real-life behavior.

After the first user test, guests were added in the dining room to make it more clear how many people the player should cook for, nevertheless, most participants still didn't take the number of eaters in consideration. In the data, a weak upward trend can be seen, see Graph 2. However, due to the small sample size, this is not of statistical significance.



Graph 3: Grams served per number of eaters

Perhaps one of the most fundamental sources of unrealistic behavior was in the recipes. People were not able to make the dishes they wanted to make, because there were only three recipes available. Because of this, comments like "I would never buy this in real life" and "This is not fair, because I put very different things in this recipe" were made. This is a big concern for the data integrity, since people are not buying what they would buy in real life. Therefore, the data isn't representative of their real-life behavior.

Some aspects of the game did generate realistic behavior though; in the survey two of the participants indicated that they never visit fresh stores, while two others claimed they often visited fresh stores. The ones who never visit fresh stores only went to the fresh stores in the game 2-3 times, while the participants who often visited fresh stores did so in game 5-7 times.

Some behaviors couldn't be measured accurately given the small sample size and the short playing time. These include expired products, the effects of the villager satisfaction, whether the weekly budget and the daily time limit are feasible once players are more familiar with the game, and in general how people behave after the first week of playing.

Data

To evaluate the effectiveness of the data collection, it needs to be seen whether the collected data helps answer the questions of the client, and how the data can be retrieved. The information the client wanted to collect was the following:

1. What has been bought + expiration date
2. What is used + expiration date
3. What is not used and when is it thrown away + expiration date
4. How much money is spent per shop visit and per week
5. How much food is prepared (for how many people)
6. Was the recipe followed? Are all ingredients present?
7. Is the preparation time correct, too short or too long?
8. How often is the catapult used?

Below are MySQL queries that answer the corresponding questions. In these queries, the username is "x". This needs to be replaced with the username of interest. The 'AND `UserID` IN (SELECT `UserID` FROM `users` WHERE `UserName` = "x")' part can also be omitted to view the information of all users, rather than a specific one.

1.

```
SELECT
`EventID`, `ProductInstanceID`, `ActionAmount`, `ActionDue`, `Day`, `Time`
FROM `eventtable` WHERE `EventType` = "moveProduct" AND
`LocationTo` = "Basket" AND `UserID` IN (SELECT `UserID` FROM
`users` WHERE `UserName` = "x")
```
2.

```
SELECT `EventID`,
`ProductInstanceID`, `ActionAmount`, `ActionDue`, `Day`, `Time` FROM
`eventtable` WHERE `EventType` = "useProduct" AND `UserID` IN
(SELECT `UserID` FROM `users` WHERE `UserName` = "x")
```
3.

```
SELECT
`EventID`, `ProductInstanceID`, `ActionAmount`, `ActionDue`, `LocationFrom`, `Day`, `Time` FROM `eventtable` WHERE `EventType` =
"wasteProduct" AND `UserID` IN (SELECT `UserID` FROM `users` WHERE
`UserName` = "x")
```
4.

```
SELECT `UserID`, `EventID`, `ActionAmount`, `Shop`, `Day`, `Time` FROM
`eventtable` WHERE `EventType` = "shopVisit" AND `UserID` IN (
SELECT `UserID` FROM `users` WHERE `UserName` = "x")
```

5.


```

SELECT `Day`, `Eaters`, `GramsServed` FROM (
SELECT `Day`, `dayStatsID` FROM `eventtable` WHERE `EventType` =
"endDay" AND `UserID` IN (
    SELECT `UserID` FROM `users` WHERE `UserName` = "x")
)AS base
INNER JOIN `daytable` ON base.dayStatsID = `daytable`.`DayStatsID`
      
```
6.


```

SELECT `Day`, `MissingProduct` FROM (
SELECT `Day`, `dayStatsID` FROM `eventtable` WHERE `EventType` =
"endDay" AND `UserID` IN (
    SELECT `UserID` FROM `users` WHERE `UserName` = "x")
)AS base
INNER JOIN `daytable` ON base.dayStatsID = `daytable`.`DayStatsID`
      
```
7.


```

SELECT `Day`, `Raw`, `Burnt` FROM (
SELECT `Day`, `dayStatsID` FROM `eventtable` WHERE `EventType` =
"endDay" AND `UserID` IN (
    SELECT `UserID` FROM `users` WHERE `UserName` = "x")
)AS base
INNER JOIN `daytable` ON base.dayStatsID = `daytable`.`DayStatsID`
      
```
8.


```

SELECT `UserID`, `EventID`, `EventType`, `ActionAmount`, `Day`, `Time`
FROM `eventtable` WHERE `EventType` = "shootCatapult" AND `UserID`
IN (
    SELECT `UserID` FROM `users` WHERE `UserName` = "x")
      
```

As demonstrated above, all the questions of client can be retrieved from the collected data. The client has access to all the data required to form models about player behavior, to ultimately gain insight in their main questions:

- Are people money driven, health driven, or convenience driven?
- How much do these factors (money, health, convenience) affect food waste?
- In which steps of the food cycle in households is food wasted, and why?

Conclusion

The goal of this project was to develop a serious game as a tool for collecting data on food waste behavior, and for improving people's food waste behavior. These goals require the game to evoke realistic actions from the player, and to make the player aware of the impact of food waste. Several ideas were conceived, and one was chosen to be developed further.

After the game had been developed, two user tests were conducted. In the second user test it became clear that the many of the measured behaviors were not representative of players' real-life behaviors. Several reasons for this have been identified:

1. Recipes are limiting player's options; they cannot make many of the recipes they would make in real life. This causes people to buy products they would otherwise not buy, and might even have no experience with. This could lead to increased food waste, which would not be representative of real-life behavior. Furthermore, the limited number of recipes also causes people to waste products that turned out not to be useable in the recipe they wanted to make.
2. Once the player has made some food waste, and has catapulted it into the village, the food waste is now visible in the village. This makes people realize there is a second goal to the game; not only should the king be kept satisfied, but the villagers too. This realization causes a change in behavior towards wasting less food. It is unknown, but unlikely, that this altered behavior is still representative of player's real-life behavior. This is problematic since the secondary purpose of this game was to improve people's food waste behavior. However, with no way of testing if real-life behavior changes, this secondary goal hinders the primary goal of collecting realistic data about food waste behavior.
3. People made many more shop visits than in real life, this is partly because it takes much less time and effort than in real life, but another factor is the fact that players do not have access to a shopping list. This forces them to either improvise in the store, or go back to the castle, to take a look in the recipe book, after which the player visits the shop yet again.
4. This game takes a while for players to grasp. Because of this, behavior is not yet realistic in the first few days of the game. This is especially concerning in combination with the second reason in this list, since this leaves only a small window wherein behavior is potentially realistic; in the beginning, behavior is unrealistic because people are still getting used to the game, but once people understand the game better, they alter their behavior to keep the villagers satisfied. This window seems to be only a few in-game days in size for most players, and the start and end of this window can differ per player, making it virtually impossible to determine whether the collected data is realistic.

Not all behaviors were unrealistic however; players who never visit fresh stores in real life also didn't visit fresh stores in the game after the first day. Players who often visit fresh stores in real life also showed this behavior in the game.

Nevertheless, it has to be concluded that the game in its current form is not a reliable tool for collecting data on food waste behavior, however, the groundworks have been laid for an improved version of the game which resolves the problems outlined in this report. The next chapter will detail what aspects of the game need to be changed to make the game successful at its intended goals.

In terms of data collection, a relational database was found to be the most suitable for the type and amount of data that this game collects. A list of events is stored in the database such that the gameplay can be reconstructed. Besides storing events, some extra information is stored as well to make it easier to retrieve information about snapshots. The client can use queries to gather insightful information about food waste behavior.

The SettingsTable serves as an input for the game; the client can change variables like the budget of the player, the duration of the day, the capacities of the storages, etc. This allows them to test different hypothesis regarding the impact of certain factors on food waste.

Future work

This chapter will describe the main aspects that need to be changed to make the game a viable tool for collecting realistic data. Additionally, a list of recommendations is given to make the game more useful and versatile. Lastly, a list of potential future use cases is given.

Recommendations

There are several crucial features that need to change in order to collect realistic data.

- Giving the player the ability to make dishes without having to follow recipes. This will greatly increase the representativeness of players' actions.
- Delaying, or even omitting the visible food waste in the village, to prevent players from optimizing their behavior to satisfy the villagers, rather than displaying their real-life behavior.
- Increase the time and/or effort it takes to go shopping. This is to prevent players from shopping way more than they do in real life. Another way to decrease the number of times people go shopping is to give them access to a shopping list, this way they won't have to go back to the castle to check the recipe again.
- Make the game easier to learn. This is very important to quickly start generating realistic data. This can be achieved by having more a more intuitive user interface and navigation. Another way is to make a tutorial that really guides the player step by step in the first day or two.

Next is a list of recommendations that are not crucial, but can further improve the game, either in terms of playability, or in terms of realistic data collection.

- There must be a solution if the player runs out of money. Currently, when this happens, the player will have to serve the king whatever products the player still has, to finish the week and receive a new budget. If the player has no money or products, the player gets stuck.
- The ability to drag products from the bag directly to the kitchen counter, rather than first having to put it in a storage.
- The ability to move a product from one storage to another, without having to put it on the kitchen counter first.
- The ability to throw away a product from the kitchen counter.
- More incentive to keep playing, e.g. by having more story in the game or by adding bonuses, these bonuses could make the food even more tasty, or help keep the village clean, or they could be purely aesthetic bonuses.
- Change the expiry date of a product if the packaging has been opened. This applies to some products (sauces) more than others (spaghetti).
- Disable or punish the ability to put products that don't belong in the freezer or fridge, in these places. Currently, all products increase in expiry date when put into the freezer, even products that really don't belong there, like milk.
- Have per-product expiry dates for the different storages. Currently all products that belong in the fridge halve their expiry date when put into the cupboard, however, in real life, some products expire much quicker than others outside of the fridge.
- There should be more tangible consequences of serving dinner too late. Currently, this just affects the king's satisfaction, however, nothing really 'happens' when the timer is full.
- It should be more clear to the player that a new day starts once the table has been cleaned. This can be done by fading the screen to black, and back to normal. Then the next guests would already sit there.

- It should be possible to save leftovers for the next day, rather than having to throw it away.
- The addition of a garbage man, to prevent the village from becoming increasingly more polluted without having the ability to clean it.
- Making it more visible that products are expired, by having an animation of a smelly scent above the product.
- Adding a talking animation for the king.
- Have a physical recipe book in the kitchen that players can click, rather than an icon.

If the game turns out to be a big success, there are some more additions that could improve it even more:

- Adding breakfast and lunch.
- Having an extra storage, namely a cellar.
- Adding ripeness to fruits, so it may take a few days after purchasing until they taste good.
- A level system wherein the time limit is dynamic; new players get more time, but once they get better, they have less time per day.
- Passwords should be encrypted.
- Eaters who can unexpectedly join or cancel.
- More UI animations and sounds
- Better systems for changing product packaging, e.g. cans vs jars vs fresh, different sizes, bonus sticker. All of these things are possible with the current game, however, they require each variation to be a unique object, even though some properties overlap.
- Allowing products and recipes to be added and edited from the server instead of from the unity editor. The challenge with this is storing the sprites in the database. They could be encoded, but then another tool would be required to encode the images.

Known bugs

These are some unintended behaviors, some of which are important to fix before using the game as a research tool. In the game code is more information about what causes the bugs. The list goes from severe, to not severe.

- If you have pans with products in them, and leave the zoomed in view of the kitchen, and then go back to the zoomed in view, the pan progress resets, and the pans duplicate, creating two progress bars, only one of which can be paused.
- Going to the cash register from the basket always leads to the supermarket cash register regardless of which shop the player is currently in.
- The euro sign is not visible in the web version of the game.
- If you drag a product to a pan, don't type anything, and drag another product to the pan, the two input fields overlap.
- The UI doesn't scale correctly if the aspect ratio is not 16x9. (The windowed version on the web is 16x9 though)
- The fresh bonus is counted per product instance instead of per product type. When using a ton of fresh products, this can cause the king's satisfaction to exceed the maximum of 5.
- The king's voice reactions after the first meal don't finish.
- In the freezer you can scroll way too far up and down.
- You can't quite scroll far enough down in the basket and the bag.
- When viewing more information about a product in the shopping basket, the 'terugzetten' button works, but the visuals don't update.

- If you are in a product section of the supermarket, and go to the cash register from the basket, and then click 'verder winkelen', you are taken back to the shop section you were in, however if you then click arrow back, you are taken back to the cash register, instead of the appropriate shop lane.
- If the game is saved and reloaded before the end of the first day, the timer will be present on the first day.
- If you save the game at the catapult or in the village, and reload the game, the catapult cost and food waste in the village aren't loaded until re-entering the scene.
- Products float above the shelves.
- The player can click on the storages in the kitchen through the recipe book.
- If you fire the catapult many times in a row, the animation stops at the wrong frame.
- The top cupboard can be opened, but not closed.

Future possibilities

The possibilities of this game aren't limited to measuring people's food waste behavior; variables can be altered to see how this behavior changes. For example, adding multiple options of the same product, with different package sizes, package types, different due dates, etc. These results can then be used to alter packaging in real life, to reduce food waste.

It can also be used to test the reception of new products. People cannot taste these through the game of course, but it can be tested whether consumers are willing to buy misshaped food like in figure 49.

Another thing that could be altered is the way in which due dates are displayed, for example, 'best before', 'use before', 'quality guaranteed until', etc. This could have an impact on how people treat products near the expiration date.

Many more variables, like budget, number of eaters, time, can all be changed find correlations between these factors and food waste, to ultimately discover what the main factors of food waste behavior are, and how to help people waste less food.

This type of technology could also be used for other fields of study, although for many research fields it is probably not worth the time and effort to create a game instead of a survey, however, for subjects where surveys have been proven to be unreliable, serious games could be a solution to gathering realistic data.



Figure 49: 'Buitenbeentjes' - misshaped fruits and vegetables sold by Albert Heijn.

References

- Allen, G. N., & March, S. T. (2006). The Effects of State-Based and Event-Based Data Representation on User Performance in Query Formulation Tasks. *MIS Quarterly*, 269-290.
- City of Surrey. (2018). *Waste sorting game*. Retrieved 7 16, 2018, from City of Surrey: <https://surrey.recycle.game/>
- FAO, G. (2011). Global food losses and food waste—Extent, causes and prevention. *SAVE FOOD: An initiative on Food Loss and Waste Reduction*.
- Food Waste Effect. (2017). *Family game*. Retrieved 7 15, 2018, from Food Waste Effect: <http://foodwasteeffect.eu/en/the-game>
- Ghost town games. (2018). *Overcooked*. Retrieved 7 15, 2018, from Ghost town games: <http://www.ghosttowngames.com/overcooked/>
- Glu games. (2016). *Cooking dash*. Retrieved 7 15, 2018, from Glu: <https://www.glu.com/games/cooking-dash/>
- Google Play. (2017). *Cooking Joy - Super Cooking Games, Best Cook!* Retrieved 7 15, 2018, from Google Play Store: <https://play.google.com/store/apps/details?id=com.biglemon.cookingjoy&hl=en>
- Györödi, C., Györödi, R., Pecherle, G., & Olah, A. (2015). A Comparative Study: MongoDB vs. MySQL. *Conference: 2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, 1-6.
- Jim Metzner Productions. (2014). *Play the zero waste game*. Retrieved 15 7, 2018, from The kids' science challenge: http://www.kidsciencechallenge.com/year-four/zw_game.php
- Mafa. (n.d.). *Supermarket shopping spree*. Retrieved from Mafa: <http://www.mafa.com/Supermarket-Shopping-Spree>
- Nordcurrent. (2014). *Cooking Fever*. Retrieved 7 15, 2018, from Nordcurrent: <http://www.nordcurrent.com/ipad/cooking-fever>
- Nukebox studios. (2017). *Food Truck Chef*. Retrieved 7 15, 2018, from Nukebox studios: <http://nukeboxstudios.com/nukebox/foodtruckchefgame/>
- Office Create. (2013). *Cooking Mama*. Retrieved 7 15, 2018, from Office Create: http://www.ofcr.co.jp/APP_CookingMama/en/
- Solid IT. (2018). *rankings*. Retrieved 7 15, 2018, from DB-engines: <https://db-engines.com/en/ranking>
- SVS Games. (2008). *Order up*. Retrieved 7 15, 2018, from SVS Games: <http://www.svsgames.com/category/projects/order-up/>
- Virtual toys. (2008). *Nintendo Wii*. Retrieved 7 15, 2018, from Virtual toys: https://web.archive.org/web/20111006124506/http://www.virtualtoys.net/index.php?option=com_content&task=blogcategory&id=14&Itemid=33&lang=en
- Wageningen University & Research. (2016). *Houdbaarheid Begrepen*. Retrieved 7 11, 2018, from <https://www.wur.nl/nl/Onderzoek-Resultaten/Onderzoeksprojecten-LNV/Expertisegebieden/kennisonline/Houdbaarheid-Begrepen.htm>

Appendices

Appendix A

Product choice

Price

Quality

Health

Freshness

Use-by-date

Discount

What do I still have?

Money available

Visitors

Product advertisements

Emotions

Weather

Season

Special occasions

Number of eaters

Preparation difficulty

Preparation time

Preparation knowledge

Variation

Eat rhythms

Allergies

Ingredients

Weekday / weekend

Diet

Weight

Calories

Principles

Religion

Habits

Luxury

Combinations

Brand

Pre-cut

Shop arrangement

Product popularity

Product in stock

Portion size

Recommendations

Doctor's advice

Packaging exterior

Packaging size

35% discount sticker

Shop choice

Service

Atmosphere

Distance to home

Distance to other shops

Payment methods

Shop advertisements

Discounts

Opening times

Cleanliness

Shop reputation

Delivery service

Business

Cooled drinks

Free coffee / tea / cookies

Freshness

Relationships with staff

Self service

Time choice

Weather

What time is it?

When should dinner be ready?

When do I have time?

When do I feel like it?

One-day discounts

Business

Frequency choice

Time

Effort

Storage size at home

Storage size of vehicle

Effort of vehicle

Speed of vehicle

Forgot a product

Planning

Storage

How much storage do I have?

How much cooled storage?

How much frozen storage?

How full are the storages?

What products do I need to add?

Does that fit?

What products can go away?

Use, move or waste?

Is product expired?

Does product still taste good?

Are there things I forgot I had?

Preparation

What do I want?

What do I have?

What had I planned?

Use everything or a bit?

Number of eaters

How much do eaters eat?

What do eaters like?

Better too much than too little or not?

Eating

I'm full

I'm not hungry

I'm in a hurry and can't finish everything

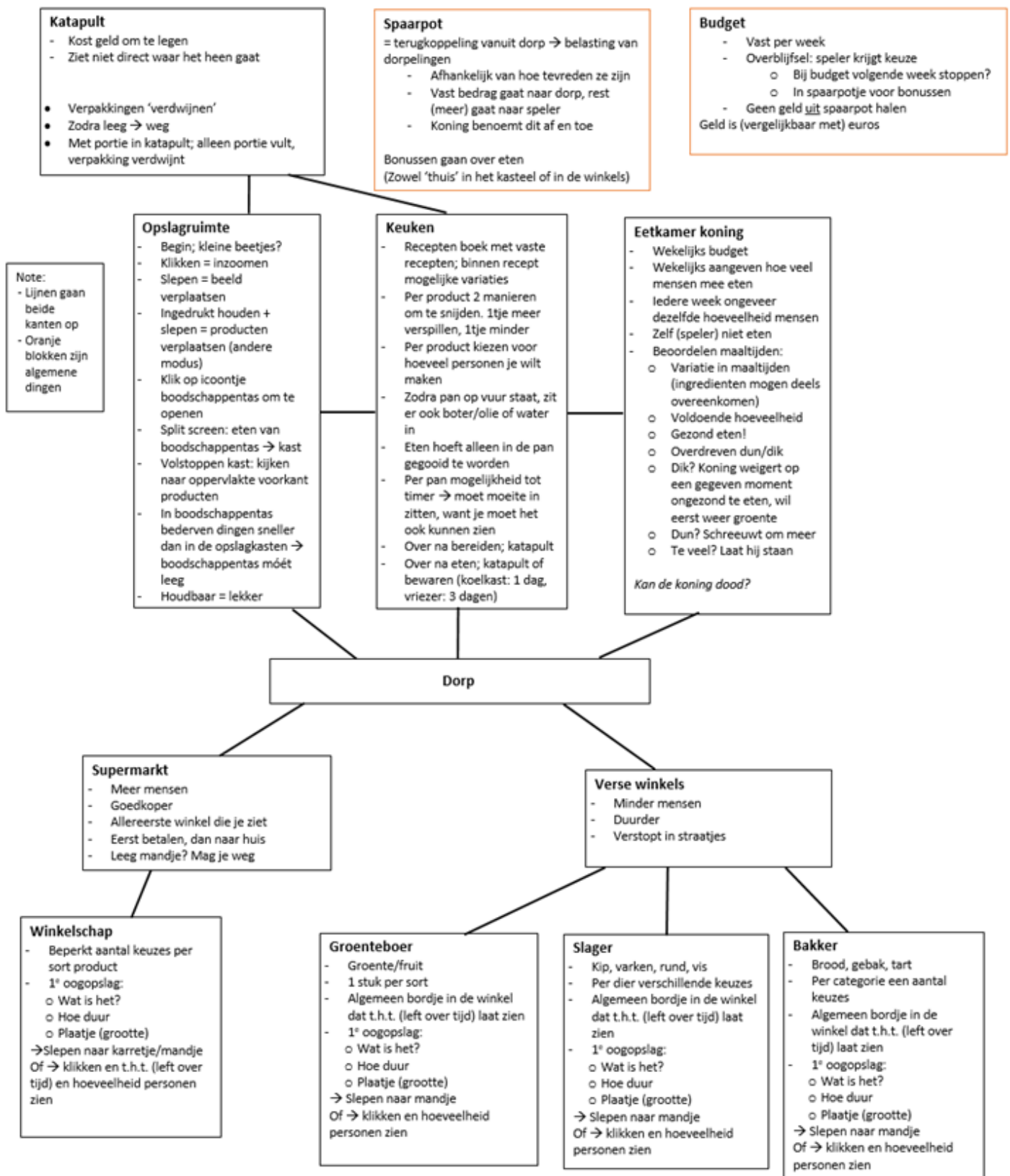
The food wasn't prepared well

An ingredient was expired

Wrong time of day

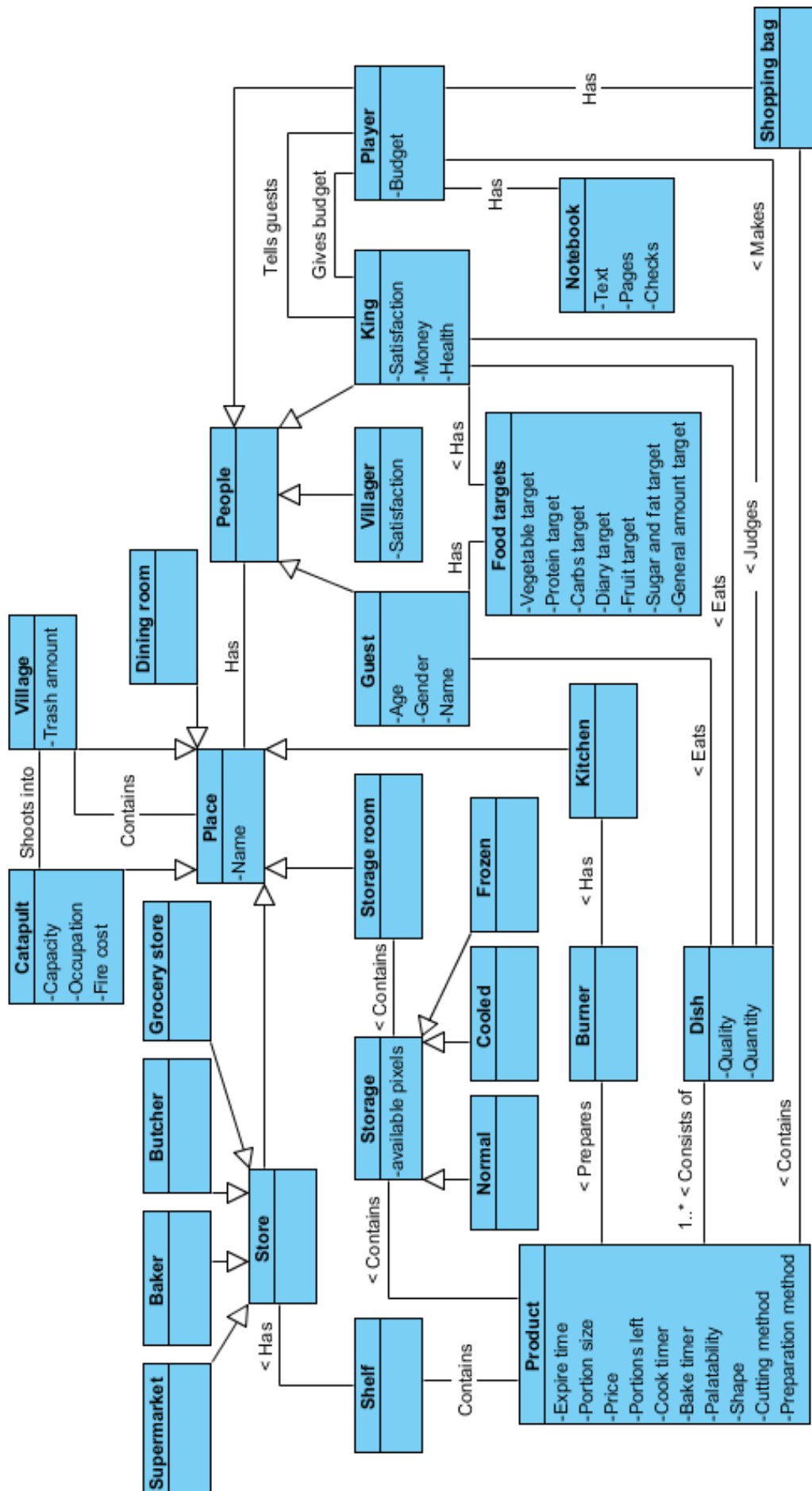
Throw away remains or save for later?

Appendix B



Appendix C

Note: this class diagram is not representative of the final game



Appendix D

Food waste game user manual

This manual consists of five parts:

1. Setting up the game on the server.
2. Instructions to give to the user.
3. Retrieving information from the server.
4. Changing settings of the game.
5. Changing other parts of the game.

Disclaimer: Passwords are not encrypted, but stored in plain text. The database is not sanitized, so it is vulnerable to SQL injection attacks. To mitigate the risk of data loss, do not host any other data in the database used for the game. Neither the game developer, nor the University of Twente are responsible for any data loss.

Setting up the game on the server

1. Upload all files in Food Waste Game\Game\PHP files to the server. These can be put in a folder to keep the server more organized.
2. On the server, create a MySQL database.
3. Import the database from Food Waste Game\Game\Database\full. Alternatively, tables can be imported individually from Food Waste Game\Game\Database\individual. When uploading individually, make sure that 'foodwastegame_database.sql' and 'foodwastegame_extra.sql' are imported last.
4. Open Food Waste Game\Game\Unity files\Food waste game\Assets\Resources\authentication\authentication.csv
5. In the first column of the csv file, replace the url with the url of the server. Make sure the filepath includes the folder that contains the PHP files uploaded in step 2.
6. In the second column of the csv file, replace the servername.
7. In the third column of the csv file replace the server password.
8. In the last column of the csv file, replace the database name.
9. Save and close the file.
10. Upload 'index.html', the 'Build' folder, the 'Build_Data' folder, and the 'TemplateData' folder located in Food Waste Game\Game\Unity files\Food waste game\Build.
11. The game should now be playable on the platforms listed at the end of this document.

Instruction for players

1. When registering, don't use a password that you use for other services. Or don't use a password at all.
2. If you want to stop playing, click the blue question mark on the top right, then click 'Opslaan en afsluiten'. This will save your progress. You can now close the browser tab. The next time you log in, you can continue playing from where you left of.

Retrieving information from the server

The tables in the database can be exported to csv.

Alternatively, parts of tables, or combined tables can be retrieved using MySQL queries. These can then also be exported to csv by clicking on 'export' in the 'Query results operations' section at the bottom.

Changing settings of the game

To change game settings like weekly budget, number of guests, capacity of storages, etc, the values in the 'settings' table in the database can be edited. The game will use the row with the highest settingID. This means that you can either edit the row with the current highest settingID, or add a new row, if you want to see the previous settings for later.

Changing other parts of the game

If properties of products, recipes, or the code of the game need to be changed, the game needs to be opened in Unity. Unity can be downloaded here: <https://unity3d.com/get-unity/download>

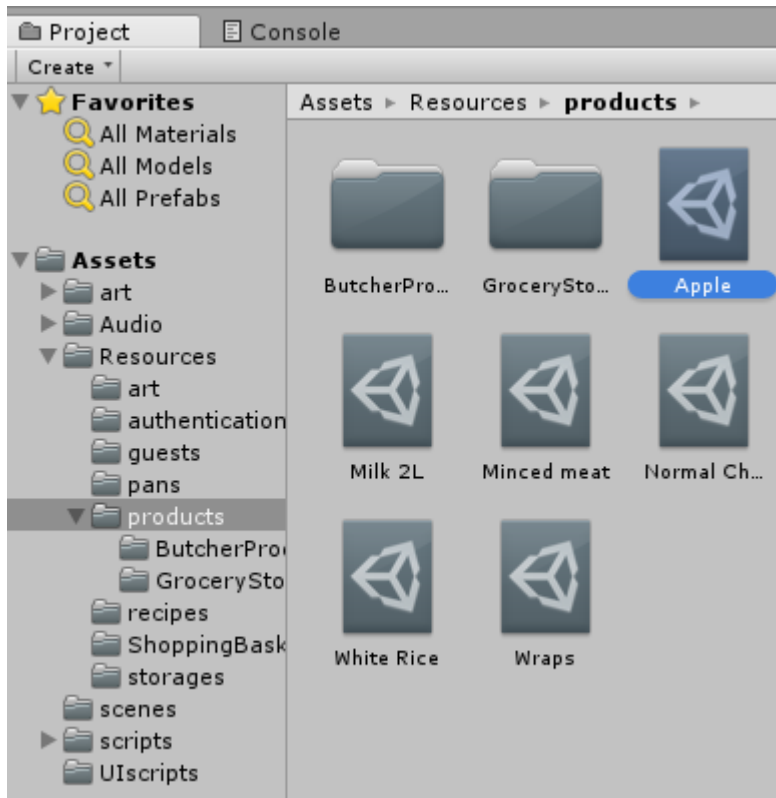
When installing, make sure to tick the WebGL box.

Next will be described how to:

- Change a product
- Add a product
- Change a recipe
- Add a recipe
- Rebuild the game to put it back on the server

Changing products

1. Products can be found under Assets -> Resources -> products. In this folder are two other folders for products in the greengrocer and butcher. Select the product that needs to be changed.

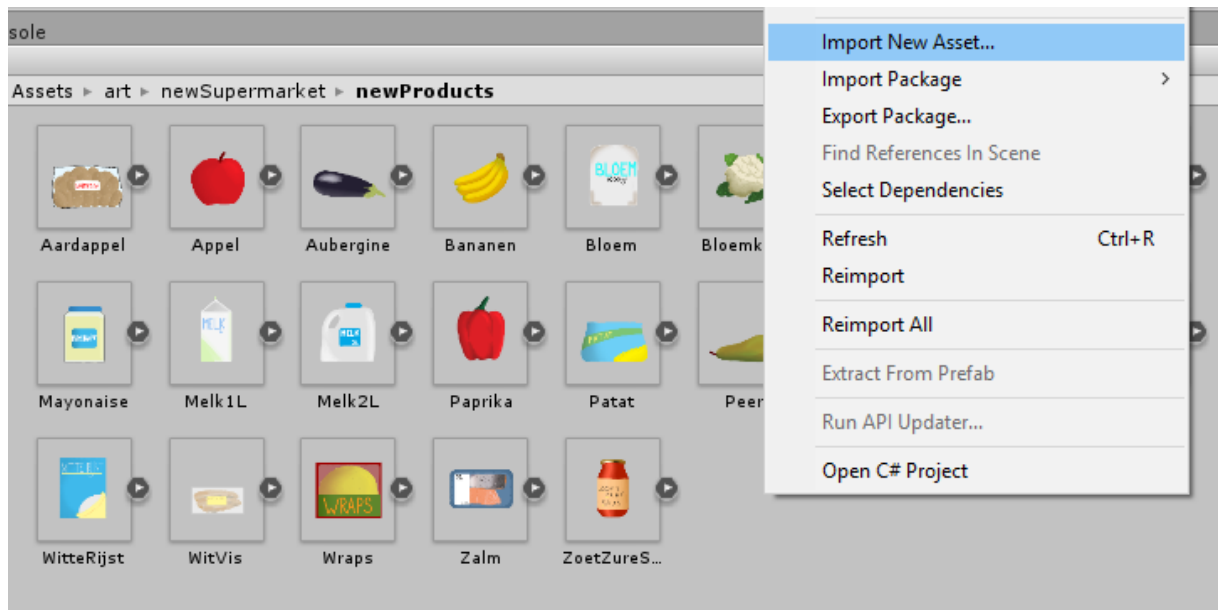


2. In the inspector, values can be changed. These are saved automatically. Most of these variables should speak for themselves, but some require some explanation.

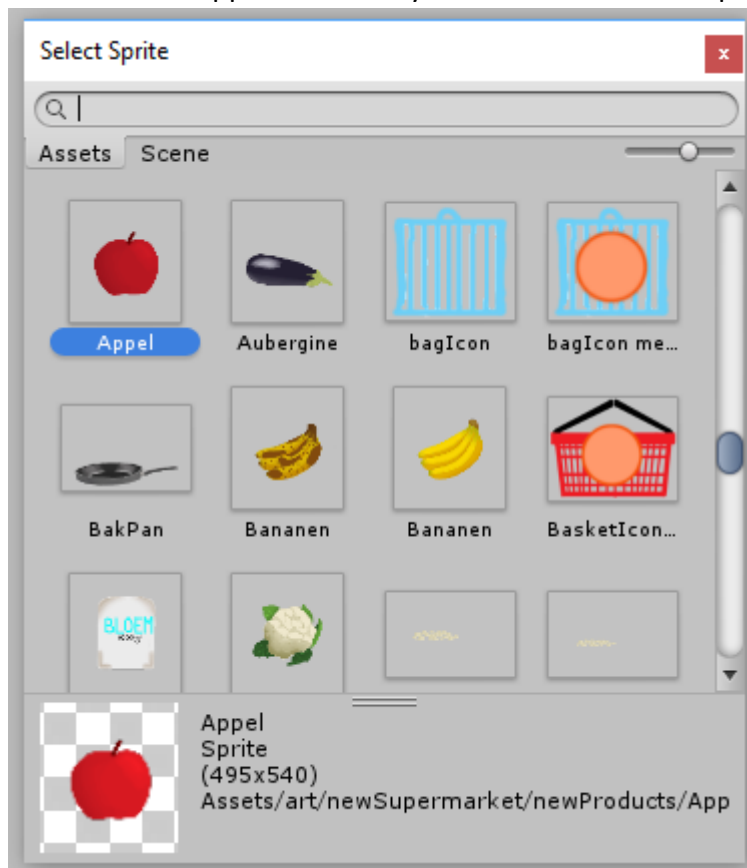


- 'due' is the number of days remaining in the selected 'Shop storage type'.
- 'shopCategory' determines in which shop section the product will be displayed.

- Sprites need to be imported into unity before they can be assigned to a product. To import a sprite, navigate to Assets -> art -> newSupermarket -> newProducts. Then right click and click 'Import new Asset...'. Select the image.

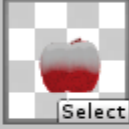


Go back to the product inspector, and click 'Select' on the image you want to change. A window will appear wherein you can search for the sprite.



- The checkboxes with pans are to allow users to put the product in that pan. Clicking a checkbox reveals three sprite slots, for the product when it is raw, done, or burnt. If you only have one sprite, fill all of these slots with that same sprite.

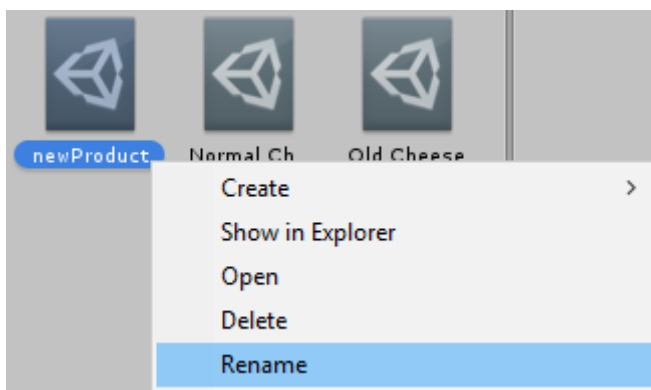
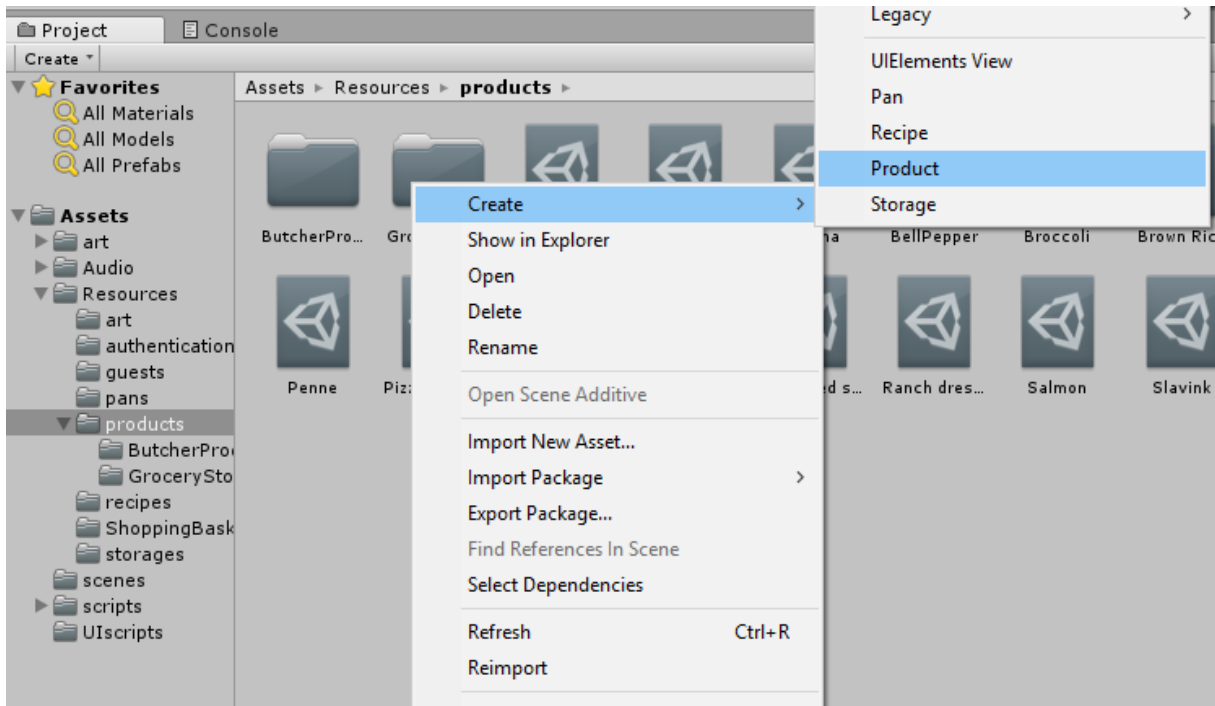
Inspector
Services
Apple
Open

name	Appel
grams	400
price	3.19
due	6
shop	Supermarket
shopCategory	Fruit
Shop storage type	Dry
Normal	
	 Select
Expired	
	 Select
bakpan	<input checked="" type="checkbox"/>
raw	
	None (Sprite) Select
done	
	None (Sprite) Select
burnt	
	None (Sprite) Select
grotepan	<input type="checkbox"/>
kleinepan	<input type="checkbox"/>
koekenpan	<input type="checkbox"/>
wokpan	<input type="checkbox"/>
proteins	0
fat	0
carbs	0
vitamins	0
minerals	0

- Proteins, fat, carbs, vitamins and minerals are currently not implemented. Changing these values does nothing as of now.

Adding a product

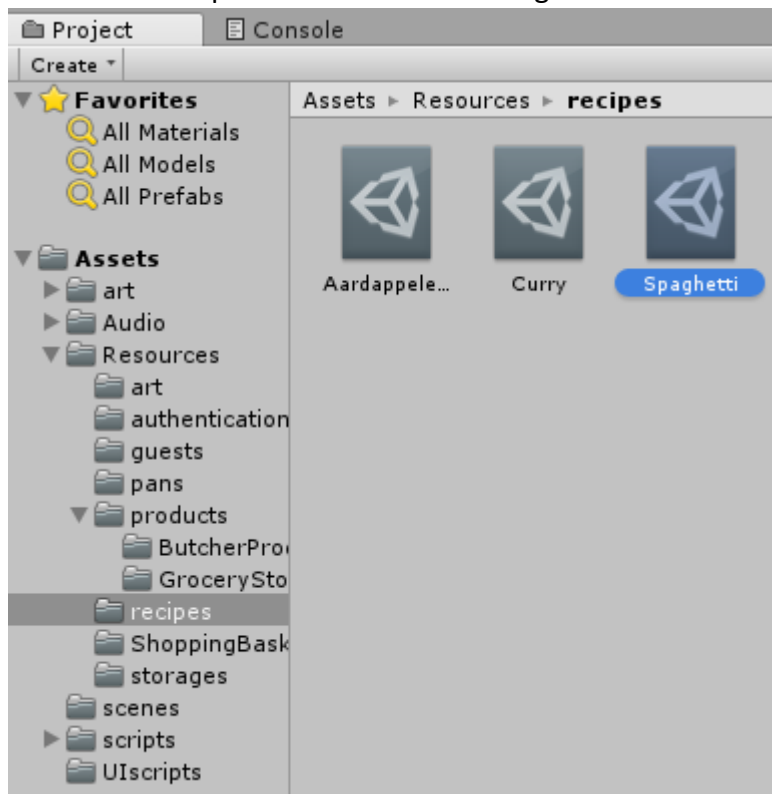
1. To add a product, navigate to the products folder (Assets -> Resources -> products).
2. Right click -> Create -> Product. A new product will be created in the products folder.
3. The name will be 'newProduct'. This name must be changed, otherwise, additional new products will overwrite any existing products named 'newProduct'.



4. If the product belongs in the butcher or in the greengrocer, drag the product to one of the two folders in the product folder.
5. The properties of the product can be edited in the inspector.

Changing recipes

1. To change a recipe, navigate to the recipes folder (Assets -> Resources -> recipes) and click on the recipe that needs to be changed.



2. In the product inspector values can be changed. These are saved automatically. The variables in the recipes require some explanation.
 - 'Sprite' is a picture of a plate with the recipe.
 - 'Unlocked' indicates whether the player has unlocked the recipe. This mechanic hasn't been tested in a long time, so it's functionality isn't guaranteed. Keep the box checked if you are unsure.
 - 'Ingredients' is a list of products that can be added to the recipe. To add a product to this list, increase the 'Size' variable, this will add an extra element (which will be a duplicate of the previous last product) Put the new product in the extra element.
 - The 'Size' variable in 'Required Pans' indicates how many different pans must at least be used for the recipe.
 - 'Pans' is a list of pans that satisfy the same function in the recipe. For example, things can be cooked in a big pan or a small pan. However, things cannot be cooked in a frying pan.
 - 'Required Products' indicates what products can be made put in this pan. Per 'Required products' there can be a list of variations of products that fulfil the same role. For example, Broccoli and Cauliflower both satisfy the role of a vegetable in the recipe. When adding products, make sure these are also in the 'Ingredient' list at the top.
 - 'Duration' indicates how many seconds it takes for that pan to go from raw to burnt.

Inspector
Services
AardappelenGroenteVlees
Open

Script

RecipeObject

Sprite

Gerecht1

Name

Aardappelen, groente, vlees

Unlocked

☒

Ingredients

Size

8

Element 0

Slavink (ProductObject)

Element 1

Potatoes (ProductObject)

Element 2

Broccoli (ProductObject)

Element 3

fresh Broccoli (ProductObject)

Element 4

fresh Potatoes (ProductObject)

Element 5

fresh Slavink (ProductObject)

Element 6

Coliflower (ProductObject)

Element 7

fresh Coliflower (ProductObject)

Required Pans

Size

3

Element 0

Pans

Size

2

Element 0

grotepan (PanObject)

Element 1

kleinepan (PanObject)

Required Products

Size

1

Element 0

Variations

Size

2

Element 0

Potatoes (ProductObject)

Element 1

fresh Potatoes (ProductObject)

Duration

40

Element 1

Pans

Size

2

Element 0

grotepan (PanObject)

Element 1

kleinepan (PanObject)

Required Products

Size

1

Element 0

Variations

Size

4

Element 0

Broccoli (ProductObject)

Element 1

fresh Broccoli (ProductObject)

Element 2

Coliflower (ProductObject)

Element 3

fresh Coliflower (ProductObject)

Duration

30

Element 2

Pans

Size

2

Element 0

bakpan (PanObject)

Element 1

koekenpan (PanObject)

Required Products

Size

1

Element 0

Variations

Size

2

Element 0

Slavink (ProductObject)

Element 1

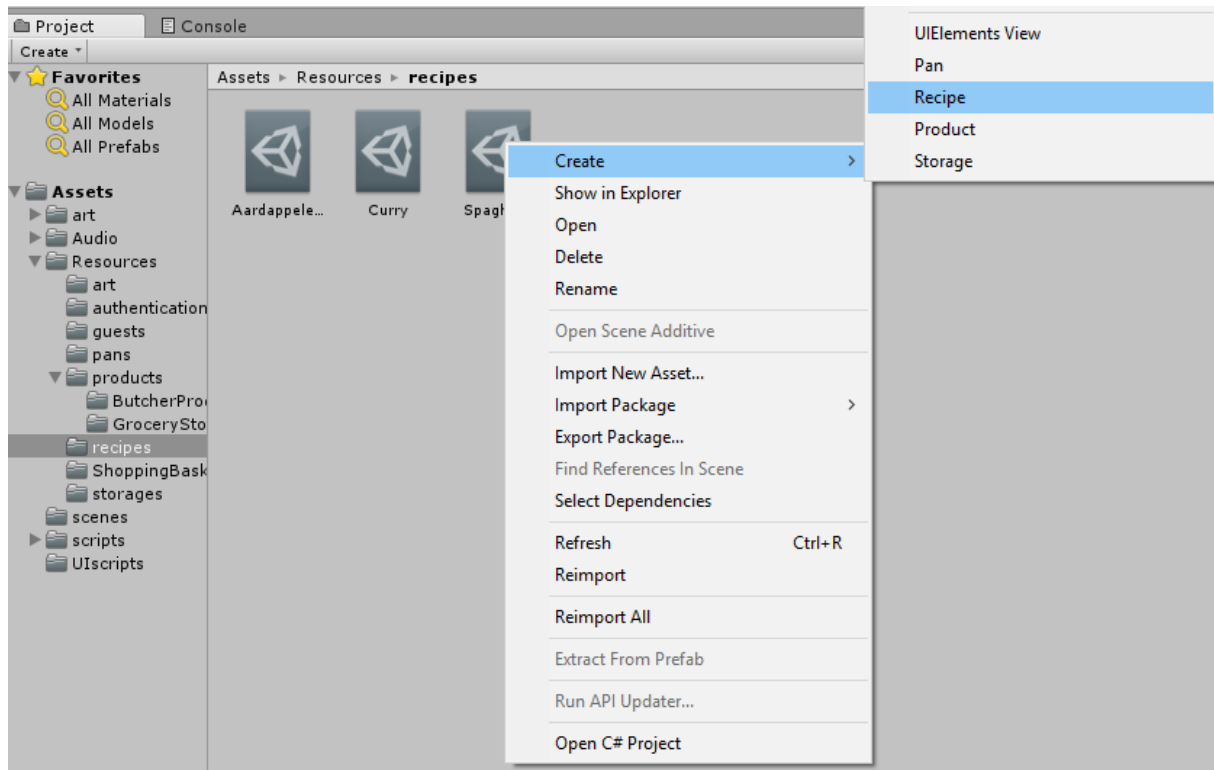
fresh Slavink (ProductObject)

Duration

30

Adding a recipe

1. To add a recipe navigate to the recipes folder (Assets -> Resources -> recipes)
2. Right click -> Create -> Recipe. The name will be 'newRecipe'. This name must be changed, otherwise, additional new recipes will overwrite any existing products named 'newRecipe'.



Change or add other things.

If you want to edit other things, like changing or adding new features, it is recommended to check out the technical document detailing the structure of the game.

Rebuild the game to put it back on the server

When the changes have been made in Unity, go to file -> build settings. Select WebGL and click 'build' at the bottom of the window. It will ask you to select a folder. Select the 'Build' folder in Food Waste Game\Game\Unity files\Food waste game.

It will then build the game for the web. This may take a while. When it is finished, 'index.html', the 'Build' folder, and the 'TemplateData' folder located in Food Waste Game\Game\Unity files\Food waste game\Build will have been updated.

Only the 'Build' folder needs to be reuploaded to the server.

Tested compatibilities

Mozilla Firefox 61.0.1	Windows 10	Works
Mozilla Firefox 56.0.2	Windows 10	Works
Safari 11.1.2	OS X	Works
Google Chrome 67.0.3396.99	Windows 10	Works
Google Chrome 67.0.3396.87	Android 8	works, but external keyboard required to type.
Microsoft Edge 42.17134.1.0	Windows 10	Works or doesn't work depending on whether 'mixed content' is allowed.
Internet explorer 11.165.17134.0	Windows 10	Works, but slow, minor UI glitches, and no audio.
Safari 5.1.7	Windows 10	Doesn't work; no WebGL support.
Google Chrome 67.0.3396.87	Android 6	Doesn't work.
Chrome 63.0.3325.152	iOS 10.3	Doesn't work
Chrome 67.0.3396.87	iOS 10.3	Doesn't work
Safari 10	iOS 10.3	Doesn't work
Safari 11	iOS 11.4	Doesn't work

Appendix E

Technical documentation

This document contains a high-level overview of the structure of the program. More detailed explanations are commented in the scripts themselves.

Main

Main is singleton. This means that there can only be one instance of this class. This instance is a static value of the class. This way it can be accessed from anywhere using Main.obj. The singleton stores many global variables that define the state of the game. These variables need to be saved in the database to save people's progress. Since Main derives from MonoBehaviour, it cannot be serialized.

That's why the 'Data' class exists. It holds a copy of the variables in main. The data class is serialized and deserialized to save data.

(An alternative may be to save all the variables in a separate column in the table, instead of saving the serialized json string in one column. This could increase backwards compatibility; with the current system, the addition or removal of a variable makes the saved serialized string incompatible with the new Data class.)

Events are stored through the addEvent method.

The shops

The supermarket contains of three scenes; lane1, lane2, and supermarketCashRegister. The lanes consist of sections each of which contains several products. When you click on a section, you will go to the shopSection scene. In here the panelLoader will load all the products in the Resources -> products folder and picks the products that match the category of that shopSection. It then loads panel prefabs to put this products in. These panels are then placed by ProductPlacer.

The butcher and greengrocer do not load the product dynamically, since every product is visible on the main screen. The player just clicks on a product and sees the infoPanel with the appropriate product. (Small note about the greengrocer, I thought 'grocery store' was a synonym to 'greengrocer', but it's actually a synonym to 'supermarket'. If you see 'grocery store' somewhere in the code or database, it refers to the greengrocer.

When a purchase is confirmed, the event is saved by calling the addEvent method in Main. It saves the store that was visited and how much was spent. (The store name is gathered from the scenename, so don't change the name unless you also change this system.)

Products

ProductObjects are scriptableObjects, these are objects not bound to any scene that can store information. They are intended as a sort of prefab, but with variables. These variables should not be changed during run time, they only act as template. The process of adding and editing products is described in the user guide.

The scripts related to these scriptableObjects are in the scripts-> product folder. To add editable variables to objects, for example a 'waste impact', it needs to be added to the ProductObject script. However, to also be able to view the new variable, it must also be added to the ProductInspector. The

productInspector overwrites the default inspector to show the variables in an organized way to the programmer.

An important distinction to make is the difference between a productObject (the scriptableObject) and a productInstance. There is only one 'spaghetti' productObject, but there may be many instances of spaghetti, each with their own due date, location, and grams left.

The 'product' class (not within the product folder) determines how products are displayed in their panel. It could be decided in the future to completely remove panels and only show the product image, this could increase the perceived realism. Either way, there are different kinds of panels in the Resources folder, for example, StoreInfoPanel, ProductChoicePanel, basketProductPanel, bagProduct, etc. These each have the product script attached. Some have more elements than others (for example, in the supermarket you can see the price, but not in the bag).

Storages

StorageObjects are also scriptableObjects. These store the properties about storages that don't change. Adding or editing a storage goes the same way as for products.

Storage is the storageInstance, this stores the properties that do change, like capacity and content.

Currently, any class that wants to add a product to a storage does so after checking whether the storage is full. The storage does not check this when adding a product. This was done because it wasn't clear what should happen if someone's bag is full, and they buy more products. (All products from basket are transferred to bag, but if that doesn't fit, what products would stay behind, would you still pay for them?) For now, the bag will just overflow. This may need to be changed.

Village

The village displays more trash, the more food waste the player has made. This is done by activating one of 6 stages of food waste, this is done in the VillageWaste class. These stages have the villager's head as a child, which activates with them.

Pans

PanObjects are also scriptableObjects. The implementation of pans is a bit poor. The content of the pan is stored in the 'pan' class, which is attached to the sprite of the pan, rather than a separate class like with products. Since MonoBehaviors are destroyed upon leaving the scene, the pan used to not be persistent. This was 'fixed', by storing and reloading the content of the pans in the 'pans' class, which also saves which pan is where. This is a bad solution however, since the progress of the pans is reset upon reloading the scene. Additionally, the pan is duplicated on reloading the scene. I recommend completely redoing the pan system, including the timers. The 'drag' class checks if a product is released on top of a storage or a pan. It may be an idea to make pans a type of storage as well.

King

When the player enters the diningRoom, the kingManager class checks if it is already known which guests will be present today. If not, it decides that in that moment. It then loads the guests.

When food is served, the properties of the dish are stored in Main. Based on these properties, the kingManager chooses the appropriate voice lines, and activates the appropriate king sprites. The king sprites include a talking animation, but it hasn't been implemented yet.

The correct plate sprites are loaded by the getPlate class which is attached to KingEating. If the king decides that the meal is not edible, he pushes the plate, this is handled by the pushPlate class attached to KingPushPlate.

If too much food was served, or the food wasn't eaten, the plates aren't empty after dinner. If everything was eaten, the plates are empty. Either way, the player must click on 'tafel afruimen'. This saves the properties of the day in the database and starts a new day.

Catapult

Food thrown away ends up in the catapult. Waste made from the storages first checks if there is space in the catapult before putting it in. Waste made from the dining room always fits in the catapult. This may not be desired, but this is to prevent the player from getting stuck if the catapult is full.

The catapult animation works in a weird way. The looping animation is disabled by default and is enabled when the 'Vuur!' button is clicked. It then disables the animation 0.98 seconds later. This is because I couldn't figure out how to display the animation only once.

These are the high-level explanations of the different mechanics, for more details about smaller classes, and specific methods, there are comments in the code that explain what they do and how they work. If things are still unclear you can email me.

Appendix F

Play test

Thank you for helping out! This is a questionnaire with a couple of questions about some actions you perform in real life. Please answer truthfully. Thank you!

* Required

General questions

First two general questions

1. What is your gender *

Mark only one oval.

☐ Female

☐ Male

☐ Other: _____

2. What is your age? *

3. Please fill in a username *

Grocery shopping

4. How many times per week do you go grocery shopping? *

Mark only one oval.

☐ One time a week

☐ Two times a week

☐ Three times a week

☐ Four times a week

☐ Five times a week

☐ Six times a week

☐ Seven times a week

☐ More than seven times a week

5. How often do you deliberately choose the cheapest option of a product? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

6. Do you pay attention to the expiration date of the product you buy? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

7. How often do you visit fresh stores, like the butcher or the grocery store? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

8. What are the main factors that decide whether you go to the supermarket or to fresh stores? *

9. What are the main factors that decide how often you go grocery shopping? *

10. What are the main factors that decide how much money you spend on food? *

Storing

11. How many products in your fridge/freezer/cupboard have specific spots? *

Mark only one oval.

- ☐ None
☐ Some
☐ Most
☐ All

12. Do products get pushed back to the shelf of the fridge/freezer/cupboard? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

13. How often do you forget about products that you still have in your fridge/freezer /cupboard? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

14. How often do products in your fridge/freezer/cupboard pass the expiration date? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

15. What are the main factors that decide if you save left-overs? *

Cooking

16. When cooking a meal, do you first finish the last bits in a package? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

17. Do you try to use as much as possible from a vegetable or fruit? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

18. What do you generally do with a vegetable or fruit that is partly rotten? *

Mark only one oval.

- ☐ Throw away entirely
- ☐ Cut out the bad parts
- ☐ Eat entirely
- ☐ Other: _____

19. Do you take your time to make a good meal? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

20. Do you follow recipes? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

21. How often do you prepare the right amount of food for the people who eat it? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

22. Do you save left-overs in Tupperware? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

23. How often do you throw away food saved in tupperware? *

Mark only one oval.

	1	2	3	4	5	
Never	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Always

24. What are the factors that decide which meal you are going to prepare? *
