Inhalt

[**Team “Toaster Roaster” stellt sich vor** 5](#_Toc387326806)

[**Das Team** 5](#_Toc387326807)

[**Das Spielkonzept** 5](#_Toc387326808)

[**Wave-Engine** 6](#_Toc387326809)

[**Plattformen** 6](#_Toc387326810)

[**Möglichkeiten** 7](#_Toc387326811)

[**Entwicklung** 7](#_Toc387326812)

[**Analytics und Advertising** 8](#_Toc387326813)

[**Kosten** 8](#_Toc387326814)

[**Real-World-Projekte** 8](#_Toc387326815)

[**Toaster Roaster – Das Konzept** 8](#_Toc387326816)

[**Die Grundidee** 8](#_Toc387326817)

[**Der Hintergrund** 8](#_Toc387326818)

[**Die Grundprinzipien** 9](#_Toc387326819)

[**Unlockables** 9](#_Toc387326820)

[**Consumables** 10](#_Toc387326821)

[Erzeugen des Projektskeletons 10](#_Toc387326822)

[Erzeugen des Desktop-Projekts 10](#_Toc387326823)

[Launcher-Projekt 11](#_Toc387326824)

[Game-Projekt 11](#_Toc387326825)

[Neue Zielplattform hinzufügen 11](#_Toc387326826)

[Projektstruktur 11](#_Toc387326827)

[**Screenflow** 12](#_Toc387326828)

[**Main Menü** 13](#_Toc387326829)

[**Game Preparation Screen** 13](#_Toc387326830)

[**Game Screen** 13](#_Toc387326831)

[**Game End Screen** 14](#_Toc387326832)

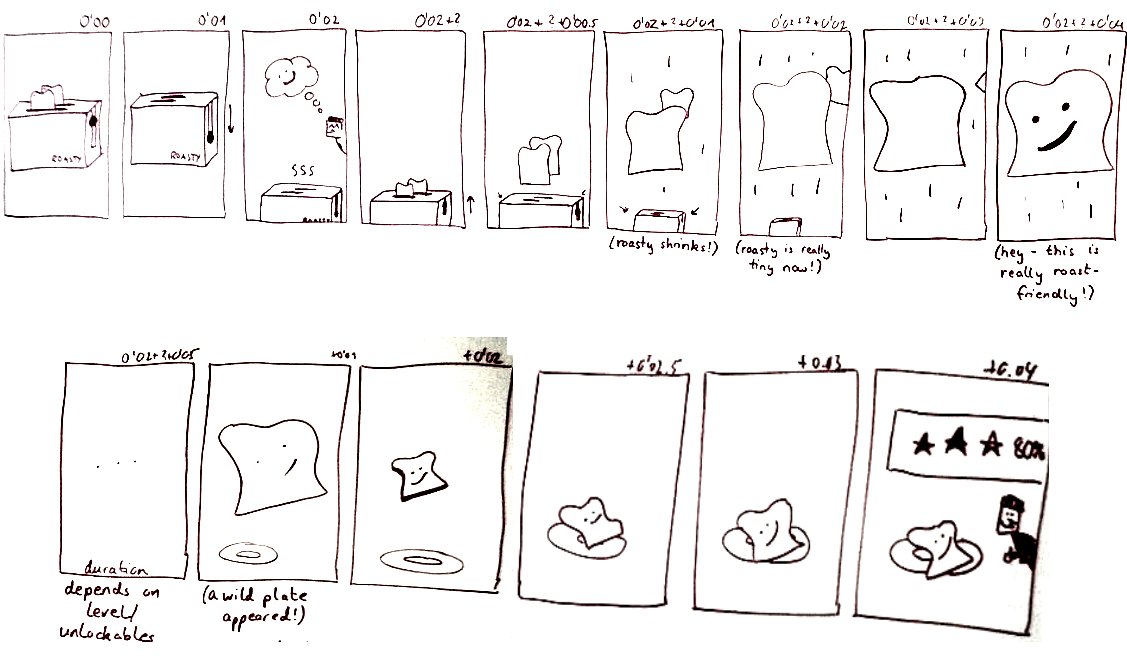
[**Statistik** 14](#_Toc387326833)

[**Highscore** 14](#_Toc387326834)

[**Hilfe** 14](#_Toc387326835)

[**Optionen** 14](#_Toc387326836)

[Personas 14](#_Toc387326837)

[Storyboard 16](#_Toc387326838)

[Source & Build Instructions 16](#_Toc387326839)

[Project structure 16](#_Toc387326840)

[Project preparation 16](#_Toc387326841)

[Prerequisites 16](#_Toc387326842)

[Android preparation 17](#_Toc387326843)

[**Architecture of Toaster Roaster** 17](#_Toc387326844)

[**Roasty and Toasty** 18](#_Toc387326845)

[Achievements 20](#_Toc387326846)

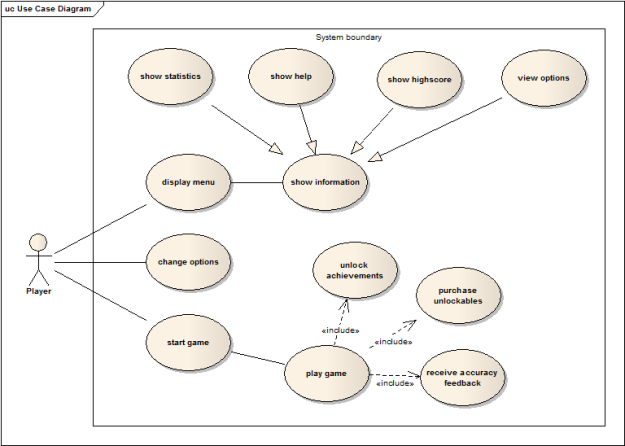
[**Mockups** 21](#_Toc387326847)

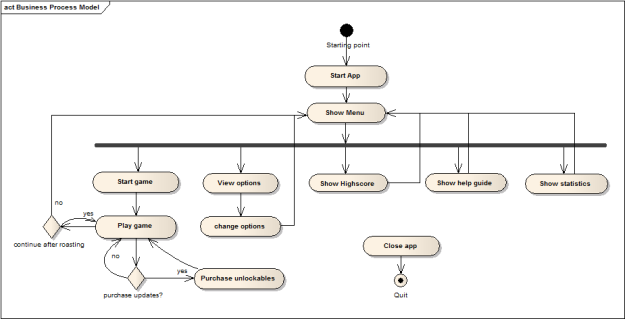
[Unlockables 24](#_Toc387326848)

[**First screenshots** 27](#_Toc387326849)

[**Vergleich der Musterübereinstimmung** 32](#_Toc387326850)

[UML diagrams 34](#_Toc387326851)

[Use case diagram (system boundary)[](http://tinf11games.files.wordpress.com/2014/04/use-case-diagram.png) 35](#_Toc387326852)

[Activity diagram[](http://tinf11games.files.wordpress.com/2014/04/business-process-model.png) 35](#_Toc387326853)

[Game Design Patterns 36](#_Toc387326854)

[Rule Patterns 36](#_Toc387326855)

[Action Patterns 36](#_Toc387326856)

[Event Patterns 36](#_Toc387326857)

[Information Patterns 36](#_Toc387326858)

[Goal Patterns 37](#_Toc387326859)

[Difficulty-Related Patterns 37](#_Toc387326860)

[Player Patterns 37](#_Toc387326861)

[**It works! Or: How we enabled roasting.** 37](#_Toc387326862)

[**Toaster Roaster Highscore Services** 38](#_Toc387326863)

[**Local highscore service** 38](#_Toc387326864)

[**Azure highscore service** 39](#_Toc387326865)

[Wave & Localytics (Signup and Basic Usage) 39](#_Toc387326866)

[Signup 39](#_Toc387326867)

[Free trial – or isn’t it? 40](#_Toc387326868)

[Configuring Localytics 40](#_Toc387326869)

[Configuring Wave 41](#_Toc387326870)

[Our first metrics 41](#_Toc387326871)

[Pages opened 42](#_Toc387326872)

[Users vs. Sessions 42](#_Toc387326873)

[Local statistics 43](#_Toc387326874)

[Achievements – Teil 1 44](#_Toc387326875)

[Implementierung 44](#_Toc387326876)

[AchievementService 44](#_Toc387326877)

[BaseAchievement 45](#_Toc387326878)

[Achievements 46](#_Toc387326879)

[Ausblick 47](#_Toc387326880)

**Team “Toaster Roaster” stellt sich vor**

[](http://tinf11games.files.wordpress.com/2014/02/logo_s.png)

Toaster-Roaster-Logo

Grüß Gott! Wir sind das Team *Toaster Roaster* und möchten uns und unser Spielkonzept an dieser Stelle kurz vorstellen.

**Das Team**

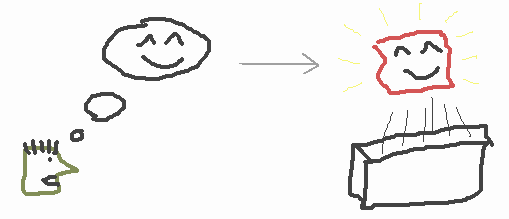
Teammitglieder von “Toaster Roaster”

Vorstellung von links nach rechts:

* Timo Frank (22) ist Toastbrot-Fan und besitzt ein Android-Smartphone
* Christian Liebel (20) liebt grün (nur die Farbe) und Windows Phone
* Simon Mayer (20) fährt gerne Ski und setzt auf Android
* Jens Ockuly (21) ist Capybara-Fan und Android-Anwender
* Manuel Rauber (23) mag Essen und ist Windows-Phone-Nutzer

**Das Spielkonzept**

Bei dieser App handelt es sich um ein Casual Game. Ausgangssituation des Spiels ist ist ein fertig getoasteter Toast, welcher aus einem Toaster katapultiert wird. Aufgrund einer unbekannten Fehlfunktion des Toasters hat der Toast keine Röstung. Ziel des Spiels ist es, nachträglich den Toast während dem Flug mit einer Röstung zu versehen, die den Anforderungen des hungrigen Menschen entspricht.

[](http://tinf11games.files.wordpress.com/2014/02/toaster.png)

Spielprinzip von Toaster Roaster

Dafür ist eine Zeitspanne gegeben, die sich durch Unlockables des Spiels erweitern lässt. Nach jedem Spiel erscheint eine Bewertung, welche sich nach der Zufriedenheit des Konsumenten richtet. Ist kein zufriedenstellendes Ergebnis erreicht, muss der Röstvorgang wiederholt werden.

Damit alle Teammitglieder das Spiel spielen können, soll es sowohl für Windows Phone als auch für Android erscheinen.

**Wave-Engine**

[](http://tinf11games.files.wordpress.com/2014/02/d4286f79f7c218b6df1c9d4cc7204f3c.png)

Wave-Logo

**Wave** ist eine 2D- und 3D-fähige Spielengine, welche die Entwicklung von Computerspielen und anderen interaktiven Grafikanwendungen auf unterschiedlichen Plattformen erlaubt (*Cross-Platform-Entwicklung*). Der Programmcode wird dabei für alle Plattformen universell in der .NET-Programmiersprache C# geschrieben. Wave basiert in Teilen auf der Cross-Platform-Lösung Xamarin. Erstmals erschienen ist die Engine im Februar 2013 und befindet sich derzeit im Beta-Stadium. Die aktuellste Version ist “Sea Lion” (1.3) vom 28. Januar 2014.

**Plattformen**

Mithilfe der Wave-Engine können Spiele für die folgenden Plattformen entwickelt werden:

* Android
* OUYA
* Mac OS X
* iOS
* Windows
* Windows Store
* Windows Phone 8

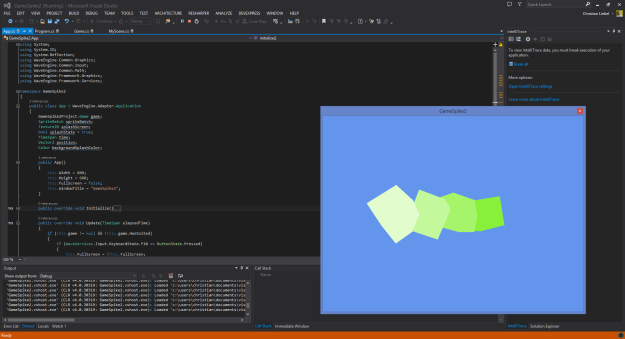
Die Bereitstellung der Spiele auf Android wird mithilfe der .NET-Implementierung für Android Xamarin.Android (vormals Monodroid) realisiert, die Bereitstellung auf iOS mithilfe von Xamarin.iOS (vormals Monotouch). Spiele für die Zielplattformen Windows und Windows Phone können direkt in Visual Studio weiterverarbeitet werden.

**Möglichkeiten**

Die Engine unterstützt sowohl 2D- als auch 3D-Spiele. Unterstützt wird unter anderem das Batching von Linien (2D, 3D) und weiteren geometrische Objekten sowie Sprites (2D), die Sichtbarkeitsprüfung [Frustum Culling](http://de.wikipedia.org/wiki/Frustum_Culling) und [Partikelsysteme](http://de.wikipedia.org/wiki/Partikelsystem). Zudem enthält Wave auch eine Physik-Engine mit Multithreadunterstützung und Kollisionserkennung sowie eine Material- und Texturdatenbank. Animationen werden ebenfalls unterstützt; das Implementieren eigener [Softwareshader](http://de.wikipedia.org/wiki/Shader) ist möglich.

**Entwicklung**

Die Entwicklung der Spiele erfolgt in der .NET-Programmiersprache C#. Als Entwicklungsumgebung kommt Microsoft Visual Studio in den Versionen 2010, 2012 oder 2013 zum Einsatz. Die Funktionalitäten der IDE einschließlich der umfangreichen Debugging-Möglichkeiten stehen dabei in vollem Umfang zur Verfügung.

[](http://tinf11games.files.wordpress.com/2014/02/d4286f79f7c218b6df1c9d4cc7204f3d.png)

Visual Studio 2013 beim Ausführen einer Testszene

Das Spiel wird zunächst für den Windows-Desktop entwickelt und ist auf diesem auch ausführbar. Später kann diese Projektmappe mithilfe des *Wave Project Converter* dann zur Veröffentlichung in plattformspezifische Projekte konvertiert werden. Zur Bearbeitung dieser plattformabhängigen Projekte ist Xamarin Studio oder die Xamarin-Erweiterung für Visual Studio erforderlich.

**Analytics und Advertising**

Die Wave-Engine enthält eigene Schnittstellen zur Analysierung des Nutzerverhaltens. Weiterhin stellt die Engine auch Schnittstellen zur Anzeige von Werbung oder zur Abwicklung von In-App-Käufen zur Verfügung. Dabei sind alle Funktionalitäten Cross-Platform-fähig.

**Kosten**

Für die Verwendung von Wave fallen keinerlei Kosten an. Es ist allerdings eine Xamarin-Lizenz erforderlich. Solange nicht mehr als 64 Kilobyte Benutzercode anfallen, kann hier allerdings auf den kostenlosen Free-Plan zurückgegriffen werden.

**Real-World-Projekte**

Das Spiel *Bye Bye Brain App-ocalipse* ([Android](http://www.amazon.com/Bye-Brain-App-ocalypse-Kindle-Edition/dp/B00AO5ICZQ) – [iOS](https://itunes.apple.com/de/app/bbb-app-ocalypse/id504136277?mt=8) – [Windows Store](http://apps.microsoft.com/windows/en-us/app/byebyebrain-app-ocalypse/d2465d5d-2ff4-4bab-9044-a2d54e5fd80f) – [Windows Phone](http://www.windowsphone.com/de-de/store/app/bbb-app-ocalypse/d9ff60f1-cece-45ab-80af-7397dd2072c2)) wurde mithilfe der Wave-Engine entwickelt und kann mittlerweile über eine Million Downloads vorweisen. Im Windows-Phone-Store erreicht die App eine Bewertung von 4,5 Sternen auf Basis von mehr als 1700 Bewertungen.

Weitere mit Wave realisierte Projekte sind die interaktive Buchbibliothek *Bookids* ([Video](http://www.youtube.com/watch?v=bj4i1GlVL80&feature=youtu.be)), die Kinderbuch-App *Three Little Pigs* sowie die Spiele *Timberwood* und *The Zombie Situation*.

**Toaster Roaster – Das Konzept**

Im vorherigen [Blogpost](http://tinf11games.wordpress.com/2014/02/14/team-toaster-roaster-stellt-sich-vor/) wurde das Konzept hinter *Toaster Roaster* schon angeschnitten. Dieser Blogpost soll die Idee bzw. das Konzept vervollständigen.

**Die Grundidee**

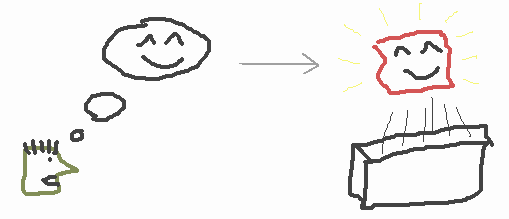
Die Grundidee hinter dem *Toaster Roaster* gestaltet sich sehr einfach: Der Spieler soll ein Muster auf einen Toast rösten, welches ihm angezeigt wird. Je genauer und schneller er dies schafft, desto mehr Punkte erhält der Spieler. Die Muster variieren von Toast zu Toast und steigen auch in den Schwierigkeitsgraden an.

**Der Hintergrund**

Alfred ist Liebhaber von frisch geröstetem Toastbrot. Ein Tag ohne seinen Toasty kann sich Alfred nicht vorstellen. Seit langem hat Alfred seinen Lieblingstoaster “Roasty”. Roasty röstet den Toast so gut, dass Alfred mit ihm ein unvergleichbares Geschmackserlebnis hat. Doch leider leidet der in die Jahre gekommene Roasty an einer unbekannten Fehlfunktion. Der fertige Toast springt aus Roasty heraus, aber es fehlt ihm die Röstung.

Für Alfred bricht im ersten Moment die Welt zusammen. Wie soll sein tägliches Toasty ohne Roasty seinen Gaumen verwöhnen? Ein Toast ohne Röstung. Für unseren Toastliebhaber Alfred unvorstellbar.

In all diesem Unglück sieht Alfred aber auch eine große Chance. Endlich hat er die Möglichkeit den Toast mit seinen Wunschmotiven zu rösten!

[](http://tinf11games.files.wordpress.com/2014/02/toaster.png)

Spielprinzip von Toaster Roaster

Ob Gesichter, Tiere oder abstrakte Formen. Alfred kann all dies umsetzen, um sein tägliches Geschmackserlebnis zu perfektionieren.

Du, als Spieler, sollst ihm dabei helfen!

**Die Grundprinzipien**

Hinter dem *Toaster Roaster* stehen ein paar Grundpinzipien, welche in das Spiel eingearbeitet werden sollen:

* **Kurzweiligkeit:**Toaster Roaster soll nicht viel Zeit in Anspruch nehmen. Es ist wichtig, dass es kurze Runden, die man mal zwischendurch spielen kann, um Wartezeiten oder ähnliches zu überbrücken.
* **Schwierigkeitsgrad:**Der Schwierigkeitsgrad soll jeweils für Kurzzeit- als auch für Langzeitspieler angemessen sein. Kurzzeitspieler sollen schnell Erfolge erreichen, während Langzeitspieler immer mehr gefordert werden.
* **Soziale Interaktion:**Soziale Interaktion ist sehr wichtig. Sei es im direkten 1:1 gegen Freunde oder über eine globale Highscore-Liste. Der Spieler soll sich jederzeit mit anderen Spielern messen können.
* **Unlockables:**Das Spiel soll verschiedene freischaltbare Elemente erhalten. Mehr Info dazu im nächsten Abschnitt.

**Unlockables**

Dem Spieler sollen verschiedene freischaltbare Elemente im Spiel zur Verfügung stehen. Die ersten Ideen sind:

* **Sprungfedern:** Kräftigere Sprungfedern schleudern den Toast höher in die Luft und geben dem Spieler mehr Zeit zum Rösten.
* **Bunsenbrenner:** Ein verbesserter Bunsenbrenner röstet den Toast schneller.
* **Aufsätze:** Aufsätze führen dazu, dass genauer geröstet werden kann.
* **Toasty:** Toasty der Toaster kann verbessert werden, sodass er schneller und genauer geröstet werden kann.
* **Toastbrot:** Es können neue Toastsorten freigeschaltet werden, die nicht so schnell Löcher in den Toast brennen.

**Consumables**

Neben Unlockables, welche permanent beständig sind, soll es auch Consumables geben, um temporär dem Spieler zu helfen:

* **Blaupause:** Auf dem Toast selbst wird das Motiv angezeigt, welches geröstet werden soll.
* **Fluxkompensator:**Dient als Zeitverzögerung, sodass man sich länger das Muster anschauen kann, welches Alfred geröstet haben möchte.

# Erzeugen des Projektskeletons

Die [Wave Engine](http://waveengine.net/) bietet sehr einfach die Möglichkeit der cross-plattform Spieleentwicklung. Dazu entwickelt man das Spiel für den Desktop-PC und hat die Möglichkeit mit HIlfe des Wave Engine Project Converters das Projekt für verschiedene Plattformen zur Verfügung zu stellen. Dabei wird wie folgt vorgegangen:

## Erzeugen des Desktop-Projekts

[](http://tinf11games.files.wordpress.com/2014/03/templates.png)

Wave Engine Project Templates

Um das Desktop-Projekt zu erzeugen, wird zuerst Visual Studio 2013 gestartet. Über das Menü kann mit “Datei” -> “Neu” -> “Projekt” ein “Wave Engine Game Project”-Template ausgewählt werden.

Das Template erstellt daraufhin eine neue Solution mit zwei Projekten:

### Launcher-Projekt

Das Launcher-Projekt dient als Container zum Laden der Wave Engine auf der jeweiligen Zielplattform. Dies bedeutet, dass für jede Zielplattform ein Launcher-Projekt existieren muss.

### Game-Projekt

Das Game-Projekt enthält den eigentlichen Code des Spiels. Hier werden alle Szenen, Entitäten etc. definiert und die Spiellogik entwickelt. Dieses Projekt ist quasi plattformunabhängig.

Leider beachtet die Wave Engine nicht das von .NET angebotene Konzept der Portable Class Libraries (PCL). Diese erzeugen echten plattformunabhängigen Code und können von verschiedenen Zielplattformen referenziert werden. Der Vorteil dieser Variante wäre, dass man nur ein einziges Projekt für die Spiellogik hat, welches gemanaged werden muss.

Die Wave Engine nutzt hier einen anderen Ansatz:

## Neue Zielplattform hinzufügen

[](http://tinf11games.files.wordpress.com/2014/03/convertertool.png)

Wave Engine Project Conveter

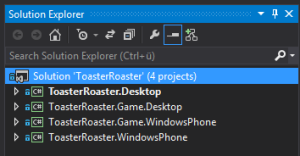
Um nun eine neue Zielplattform hinzuzufügen, muss der Wave Engine Project Converter geöffnet werden. Nach der Auswahl des Pfades zum Referenz-Projekt kann angewählt werden, in welche Zielplattformen das Projekt übersetzt werden soll.

Der Converter erzeugt nun für die jeweiligen Plattformen ein Launcher- und ein Game-Projekt.\*

Der Launcher selbst beinhaltet plattformspezifischen Code für die jeweilige Zielplattform.

Das Game-Projekt beinhaltet ein Projekt, welches die jeweilige Zielplattform anspricht, allerdings werden die Source-Dateien vom Referenz-Projekt eingebunden. (Mit PCL gäbe es weiterin nur ein einziges Game-Projekt, welches von den Launchern referenziert wird.)

## Projektstruktur

[](http://tinf11games.files.wordpress.com/2014/03/projektuebersicht.png)

Projektübersicht

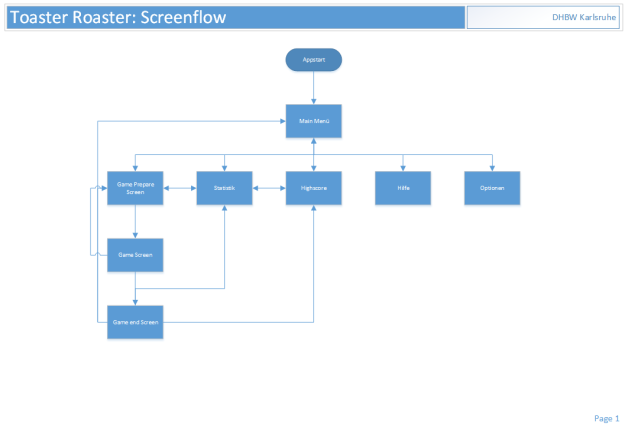
Nach dem die Projekte entsprechend angelegt wurden, haben wir nun folgende Projektstruktur:

* ToaterRoaster.Desktop: Launcher-Projekt für den Windows Desktop-Client
* ToasterRoaster.WindowsPhone: Launcher-Projekt für den Windows Phone 8 Client
* ToasterRoaster.Game.Desktop: Game-Projekt für den Windows Desktop-Client
* ToasterRoaster.Game.WindowsPhone: Game-Projekt für den Windows Phone 8 Client

Die beiden Game-Projekte referenzieren an dieser Stelle den gleichen Source-Code. Das bedeutet, dass die Spiellogik nur einmal geschrieben werden muss und in allen Zielplattformen verwendet wird.

**Screenflow**

Mit diesem Beitrag soll der Screenflow erläutert werden.

[](http://tinf11games.files.wordpress.com/2014/03/screenflow1.png)

ToasterRoaster Screenflow

Wenn die App gestartet wird, landet der Spieler im Main Menü.

**Main Menü**

Von hier aus hat er folgende Möglichkeiten:

* Spiel starten: Anzeige Game Preparation Screen
* Statistik: Anzeige Statistk
* Highscore: Anzeige Highscore
* Hilfe: Anzeige Hilfe
* Optionen: Anzeige Optionen
* Spiel beenden: Spiel wird beendet

**Game Preparation Screen**

Auf dieser Seite werden Informationen (z.B. aktuelle Upgrades, Consumables, das Motiv) zum nächsten Spiel dargestellt.

Von hier aus kann der Spieler entweder das Spiel starten, die Statistik oder das Hauptmenü aufrufen.

**Game Screen**

Hier wird das eigentliche Spiel gespielt. Er hat die Möglichkeiten durch ein “Retry” zurück auf den Game Preparation Screen zu gelangen oder das Spiel zu beenden. Ist das Motiv nachgezeichnet, landet er automatisch beim Game End Screen.

**Game End Screen**

Dem Spieler wird hier angezeigt, wie gut er das Motiv nachgezeichnet hat, ob er Consumables gesammelt hat und wie viel Spielgeld er erspielt hat.

**Statistik**

Anzeige der eigenen Spielstatistik. Beispiele:

* Gespielte Spiele
* Gewonnene Spiele
* Verlorene Spiele
* Genauigkeit
* Erspieltes Geld
* Welche Achievements erreicht wurden

**Highscore**

Anzeige einer globalen Highscore-Liste mit Suche/Filterung nach Freunden.

**Hilfe**

Anzeige der Hilfe, wie das Spiel gespielt wird.

**Optionen**

Diverse Optionen wie Lautstärke der Musik, Geräusche oder Zurücksetzen des lokalen Spielstandes.

# Personas

In this article, we want to present the [personas](http://en.wikipedia.org/wiki/Persona#In_user_experience_design) our game tries to address.

[](http://tinf11games.files.wordpress.com/2014/03/leonjaques.png)Leon-Jaques (8) is a very active boy. For this year’s summer vacation, his parents decided to get him a smartphone. However, they want him to play educational games. Toaster Roaster is a casual, yet educational game encouraging young people to be creative and engage in creative processes. Because of its levels of increasing difficulty, the player’s skills increase over time.

[](http://tinf11games.files.wordpress.com/2014/03/hubert.png)Hubert (53) is an engineer at a large industrial corporation who has lots of spare time. When he arrives at home after a long working day, he looks for a special intellectual challenge which he finds in Toaster Roaster. Due to the thrilling and fetching gameplay of Toaster Roaster and the ability to achieve a better highscore using consumables and unlockables, Hubert is really addicted to the game.

[](http://tinf11games.files.wordpress.com/2014/03/svetlana.png)Светлана (22) is a student from Russia who currently is on a students exchange in Europe. She only knows the kyrillic alphabet and isn’t able to read the latin characters yet. In order to get to know the foreign glyphs, she uses Toaster Roaster. The integration of social media and leaderboards improve the player’s skills and therefore support the learning process.

# Storyboardhttp://tinf11games.files.wordpress.com/2014/03/storyboard.png

# Source & Build Instructions

In this article we want to show you how to get the code and how to compile it.

You can get the source at GitHub.

## Project structure

The project contains two types of projects:

* Launcher project: Platform specific project to start up the game on its target platform
* Game project: Platform independent project (PCL) containing the game logic

The solution contains currently the following projects:

* ToasterRoaster.Desktop: Desktop launcher project
* ToasterRoaster.WindowsPhone: Windows phone launcher project
* ToasterRoaster.Android: Android launcher project
* ToasterRoaster.Game.\*: Game logic projects for the specific platform to use the correct WaveEngine references. The projects share code.

## Project preparation

### Prerequisites

* Visual Studio 2013
* WaveEngine (<http://waveengine.net>)
* Xamarin.Android (<http://xamarin.com>) (only needed, if you want to compile the Android 4.4)
* Windows Phone SDK (<http://dev.windowsphone.com/en-us/downloadsdk>) (only needed, if you want to compile for Windows Phone)
* HyperV (if you want to use the Windows Phone Emulator; will be installed with Windows Phone SDK)

### Android preparation

If you want to compile for Android you need to install Xamarin.Android. After the installation, start the Android SDK Manager and download the following packages for Android 4.4.2 (API 19):

* SDK Platform
* ARM EABI v7a System Image

After the download, open the Android AVD Manager and create a new virtual device with the following setup:

* Device: Use the one you want, should have at least 1 GB RAM
* Target: Android 4.4.2 – API 19
* CPU/ABI: ARM (armeabi-v7a)
* Skin: HVGA
* Use Host GPU: yes

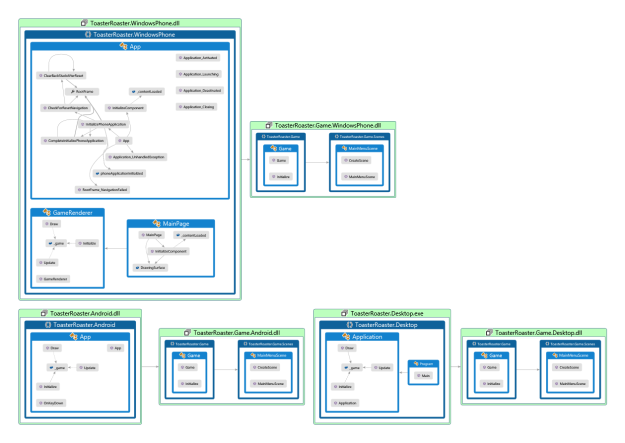
You now can compile for Android and launch the Emulator for testing purposes.

**Architecture of Toaster Roaster**

In this blog post we want to explain Toaster Roaster’s architecture on the basis of its architecture diagram. Currently, you see the stage of development as of today. This only contains one scene, the main menu scene, at the moment. In addition, we want to explain the concepts regarding the cross-platform ability, [which we described before](http://tinf11games.wordpress.com/2014/03/05/erzeugen-des-projektskeletons/).

[The Wave engine](http://tinf11games.wordpress.com/2014/02/15/wave-engine/) supports the development of game apps for different platforms. This is the usual workflow: The game is developed within Visual Studio and debugged as a Windows desktop application. This project is referred to as *core project*.

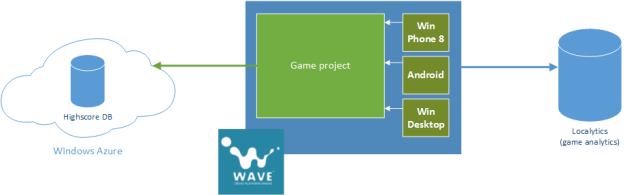
When the game is to be tested on one of the target platforms by using it on a real device or emulator, you have to convert it first using Wave’s project converter for the specific target platform. The resulting project contains copies of the source code files from the core project. Unfortunately, this project is inflexible, because you have to export a new target project for each new state of development. To get around this and get the latest state of development from the core project when compiling the platform-specific project, we used the so called [code sharing feature](http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj714082%28v=vs.105%29.aspx). Using this feature you can link source code files from other projects to several projects, each in one and the same stage of development.

[](http://tinf11games.files.wordpress.com/2014/03/architecturecodemap.png)

Code-Map of Toaster Roaster

The Wave engine additionally supports game analytics using the mobile app marketing service [Localytics](http://www.localytics.com). However, this system boundary is handled by Wave, and the game developer does not have to care about the connection.

The high scores are stored in a database hosted by Microsoft’s cloud platfrom [Windows Azure](http://www.windowsazure.com). This system boundary has to be managed by the game developer, as it is not provided by the game engine.

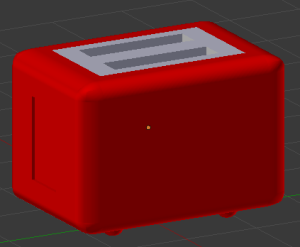
[](http://tinf11games.files.wordpress.com/2014/03/networkarchitecture1.png)

**Roasty and Toasty**

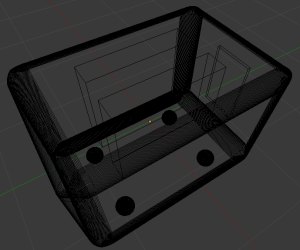
In this post I will explain the procedure for creating the first 3D concepts. For the 3D objects I used the graphics application Blender 2.69 under windows.

I think a step by step text instructions for creating the toaster would be very boring. So I took a little video while creating the 3D objects in Blender.

The Roasty default template does not have a “roast-button” and no toasts.  
Picture of the Roasty template with materials (color: red):

[](http://tinf11games.files.wordpress.com/2014/03/toaster_template.png)

Picture with all edges and verts.

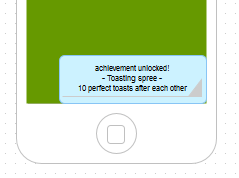
[](http://tinf11games.files.wordpress.com/2014/03/toaster_z.png)

The final picture of Roasty with two toasts and a roasty button on the left side for this week.

[](http://tinf11games.files.wordpress.com/2014/03/toaster_show_14_03_2014.png)

# [Achievements](http://tinf11games.wordpress.com/2014/03/19/achievements/)

Achievements are small goals, which can be reached during gameplay. After reaching each goal the user will be noticed via popup. Achievements are ought to extend the user game experience.

[](http://tinf11games.files.wordpress.com/2014/03/achievements-mockup.png)

**The following goals are intented:**

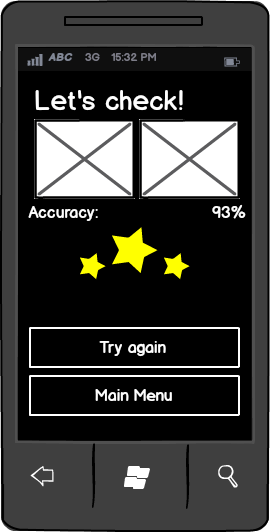
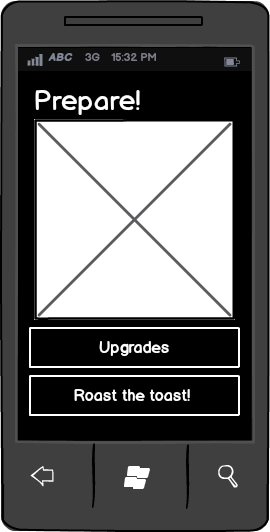
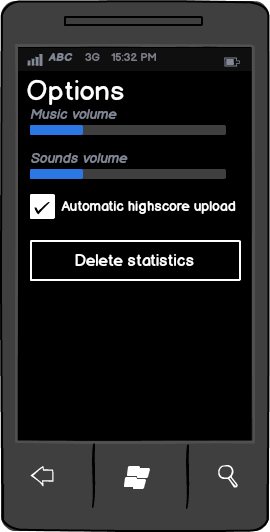
* Basic achievements

1. Go Roast a Toast! – first time starting a single player game
2. Who’s the bread-champ? – first time winning a multiplayer game
3. Interesting isn’t it? – reading about the team

* Gameplay

1. Nice Toast! – first time making a perfect toast
2. Go buy some new Toast! – first time buying a better Toaster
3. Give me Power Edison! – first time buying a better Toaster
4. Toasting spree – 10 perfect toasts after each other
5. Like a ToaPro – 20 perfect toasts after each other
6. You get the idea – toasted 10 toasts
7. The more the better – toasted 50 toasts
8. I love bread – toasted 100 toasts
9. Hurt me toasty – toasted 500 toasts
10. Please no more – toasted 1000 toasts
11. Warp 7 Mr. Toast! – toasted within 1 seconds
12. Manipulating time! – toasted within 0,5 second

[**Mockups**](http://tinf11games.wordpress.com/2014/03/20/mockups/)

This post will show our mockups. They will follow the [Screen flow diagram](http://tinf11games.wordpress.com/2014/03/06/screenflow/).[](http://tinf11games.files.wordpress.com/2014/03/mainmenu.png)[](http://tinf11games.files.wordpress.com/2014/03/game-end-screen.png)[](http://tinf11games.files.wordpress.com/2014/03/game-prepare-screen.png)[](http://tinf11games.files.wordpress.com/2014/03/highscore.png)[](http://tinf11games.files.wordpress.com/2014/03/statistics.png)[](http://tinf11games.files.wordpress.com/2014/03/options.png)

# Unlockables

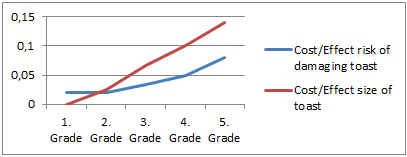
Unlockables are things which can modify the game experience. By increasing the game difficulty, improvements during gameplay are necessary to be able to achieve goals. Unlockables can be afforded to improve several components in the game. The player can get them in a **store**.

* **Unlockable:** Name of the unlockable
* **Value:** Costs of this unlockable
* **Effect:** The improvement this unlockable will create

**Toasts:**

|  |  |  |
| --- | --- | --- |
| **Unlockable** | **Value** | **Effect** |
| Toast (1. Grade) | 100 | * Risk of damaging toast in roasting process: -2% |
| Toast (2. Grade) | 200 | * Risk of damaging toast in roasting process: -4% * Size of toast: +5% |
| Toast (3. Grade) | 300 | * Risk of damaging toast in roasting process: -10% * Size of toast: +20% |
| Toast (4. Grade) | 400 | * Risk of damaging toast in roasting process: -20% * Size of toast: +40% |
| Toast (5. Grade) | 500 | * Risk of damaging toast in roasting process: -40% * Size of toast: +70% |

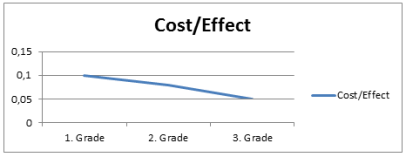
**Toasts value benefit:**

[](http://tinf11games.files.wordpress.com/2014/03/benefit1.png)

**Toaster:**

|  |  |  |
| --- | --- | --- |
| **Unlockable** | **Value** | **Effect** |
| Toaster (1. Grade) | 300 | * More time to roast the toast: +30% |
| Toaster (2. Grade) | 1000 | * More time to roast the toast: +80% |
| Toaster (3. Grade) | 3000 | * More time to roast the toast: +150% |

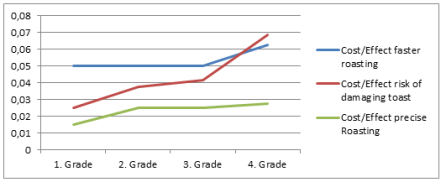
**Toaster value benefit:**

**[](http://tinf11games.files.wordpress.com/2014/03/benefit2.png)**

**Roasting fire:**

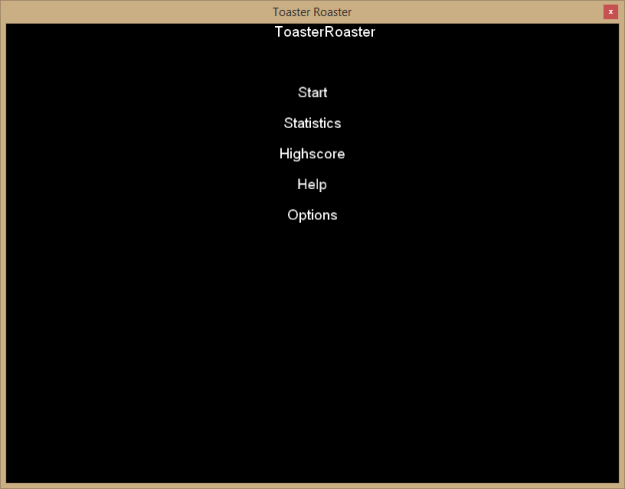
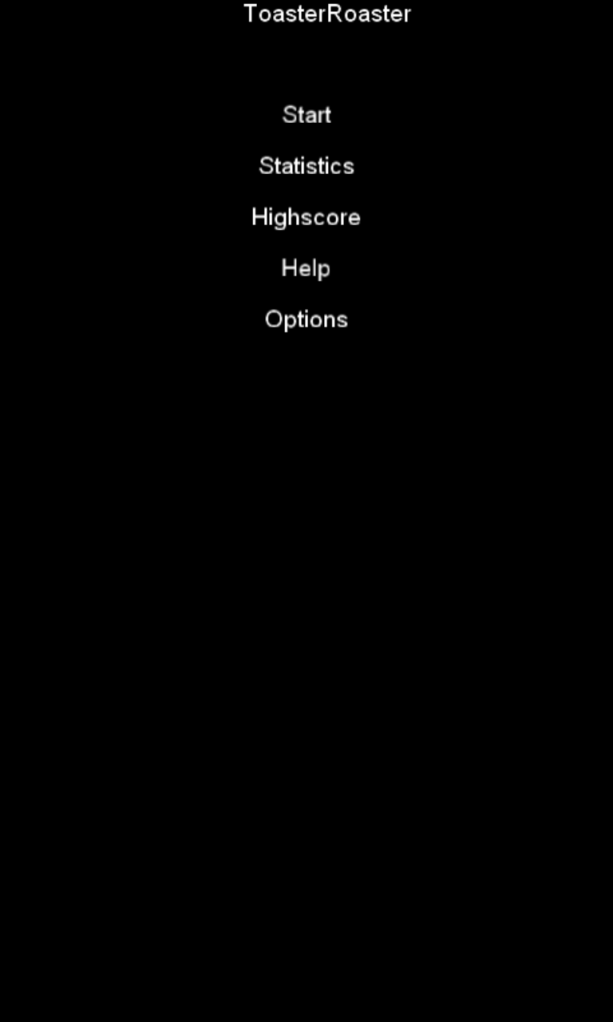
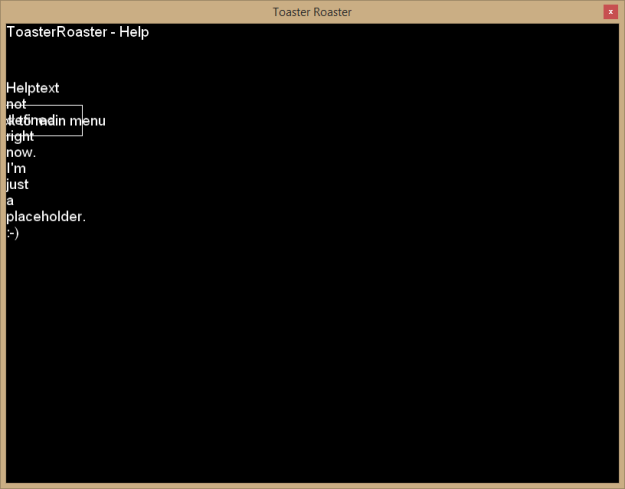
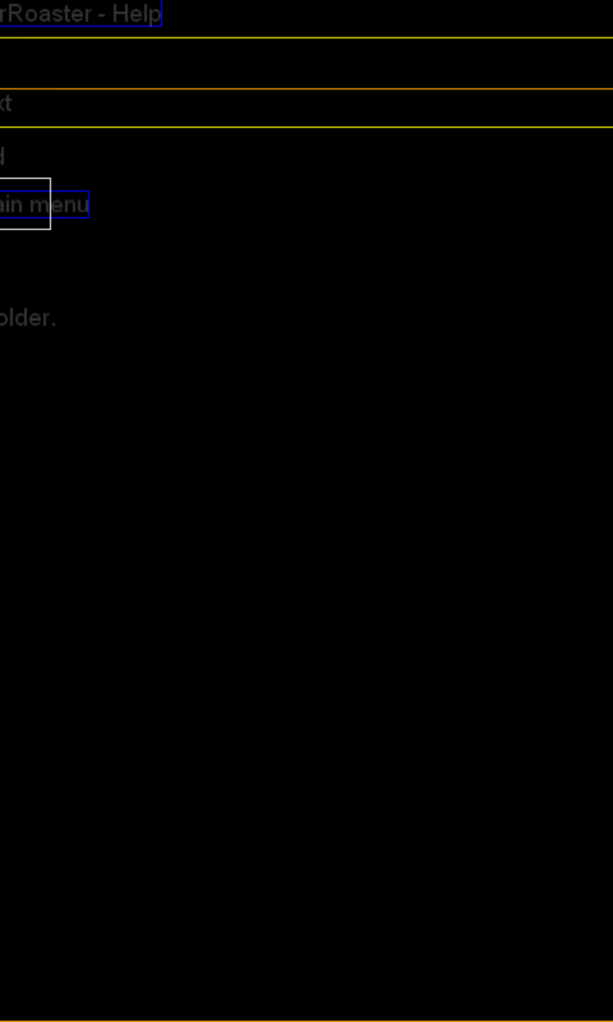
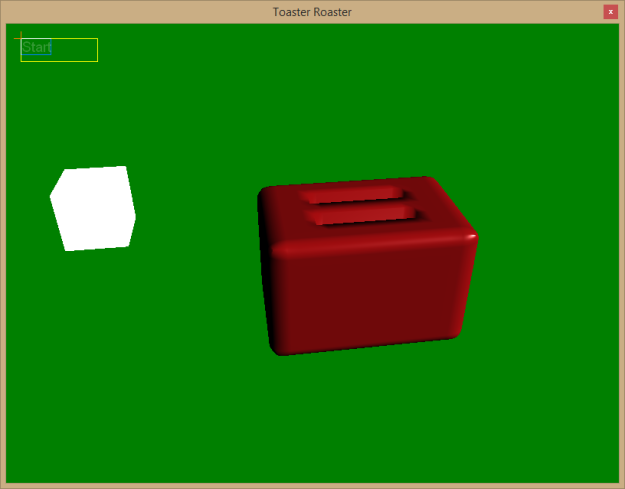
|  |  |  |
| --- | --- | --- |
| **Unlockable** | **Value** | **Effect** |
| Roasting fire (1. Grade) | 200 | * Faster Roasting: +10% * Risk of damaging toast in roasting process: +5% * Precise Roasting: +3% |
| Roasting fire (2. Grade) | 400 | * Faster Roasting: +20% * Risk of damaging toast in roasting process: +15% * Precise roasting: +10% |
| Roasting fire (3. Grade) | 600 | * Faster Roasting: +30% * Risk of damaging toast in roasting process: +25% * Precise roasting: +15% |
| Roasting fire (4. Grade) | 800 | * Faster Roasting: +50% * Risk of damaging toast in roasting process: +55% * Precise roasting: +22% |

**Toaster Value benefit:**

[](http://tinf11games.files.wordpress.com/2014/03/benefit3.png)

###### (additions could be possible)

**First screenshots**

In this post you will see some screenshots of a very very first alpha test, running on Desktop and Windows Phone 8.[](http://tinf11games.files.wordpress.com/2014/04/mainmenu.png)[](http://tinf11games.files.wordpress.com/2014/04/mainmenu_wp.png)[](http://tinf11games.files.wordpress.com/2014/04/help.png)[](http://tinf11games.files.wordpress.com/2014/04/help_wp.png)[](http://tinf11games.files.wordpress.com/2014/04/game.png)

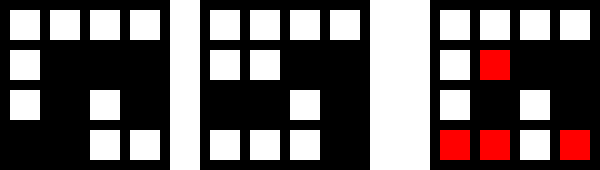
**Vergleich der Musterübereinstimmung**

[Kommentar hinterlassen](http://tinf11games.wordpress.com/2014/04/04/vergleich-der-musterubereinstimmung/#respond)

Das zentrale Spielelement von Toaster Roaster ist die Berechnung der Übereinstimmung des vom Nutzer gemalten Muster mit dem durch *Alfredo* vorgegeben Musters. In der ersten Version von Toaster Roaster wird dabei auf eine recht naive Implementierung gesetzt:

Beide Muster werden als eindimensionale Bitmap aufgefasst. Ein gezeichneter (schwarzer) Pixel wird dabei mit dem Wert “1″ (true) versehen, ein nicht-gezeichneter (weißer) Pixel mit dem Wert “0″ (false). Intern werden beide Bitmaps in Form eines zweidimensionalen Boolean-Arrays vorgehalten.

Zum Vergleich wird die Übereinstimmung beider Muster Pixel für Pixel in Prozent berechnet. Die erwarteten Übereinstimmungswerte sind allerdings relativ gering, da der Benutzer schon sensumotorisch das Bild nicht exakt nachzeichnen kann. Die Werte sinken mit zunehmender Auflösung.

[](http://tinf11games.files.wordpress.com/2014/04/patternmatching.png)

Mustervergleich

Die obige Abbildung zeigt das Berechnungsverfahren. Auf der linken Seite ist das vorgegebene Bild zu sehen sowie die Benutzereingabe, auf der rechten Seite das Matching: Grundlage für die Berechnung ist die Anzahl aller Felder (4 x 4 = 16). Als korrekt gezeichnete Felder werden die Pixel aufgefasst, bei denen der Zustand übereinstimmt. Im Beispiel sind dies 12 Felder, 4 Felder sind nicht korrekt gezeichnet (rot dargestellt) worden. Die Genauigkeit beträgt also 12 : 16 x 100 = 75%.

Nachstehend wird der Quellcode der Vergleichsmethode in C# dargestellt.

public class BoardComparer

    {

        /// <summary>

        /// This method returns the accuracy of the redrawn pattern in percent.

        /// </summary>

        /// <param name="array1">One byte array containing either the user-drawn or the computer-drawn pattern.</param>

        /// <param name="array2">One byte array containing either the user-drawn or the computer-drawn pattern.</param>

        /// <returns>The accuracy of the redrawn pattern in percent</returns>

        public double GetAccuracyInPercent(bool[,] array1, bool[,] array2)

        {

            var totalCells = array1.GetLength(0) \* array1.GetLength(1);

            var matchingCells = 0d;

            if (!(array1.GetLength(0) == array2.GetLength(0) && array1.GetLength(1) == array2.GetLength(1)))

            {

                throw new InvalidOperationException("The bounds of the arrays do not match.");

            }

            for (var xPosition = 0; xPosition < array1.GetLength(0); xPosition++)

            {

                for (var yPosition = 0; yPosition < array1.GetLength(0); yPosition++)

                {

                    if (array1[xPosition, yPosition] == array2[xPosition, yPosition])

                    {

                        matchingCells++;

                    }

                }

            }

            return (matchingCells / totalCells \* 100);

        }

    }

Da es sich hierbei, wie oben angemerkt, um eine Kernfunktion des Spiels handelt, muss ihre Richtigkeit gewährleistet sein. Aus diesem Grund wird die Methode unter Zuhilfenahme von Unit-Tests getestet:

    [TestMethod]

    public void Spielfeldvergleich\_funktioniert()

    {

        var sut = new BoardComparer();

        var array1 = new bool[4, 4];

        var array2 = new bool[4, 4];

        array1[0, 0] = true;

        array1[0, 1] = true;

        array1[0, 2] = true;

        array1[0, 3] = true;

        array1[1, 0] = true;

        array1[1, 1] = true;

        array1[1, 2] = true;

        array1[1, 3] = true;

      var accuracy = sut.GetAccuracyInPercent(array1, array2);

        Assert.AreEqual(accuracy, 50.0);

# [UML diagrams](http://tinf11games.wordpress.com/2014/04/04/uml-diagrams/)

Newly created use case diagram & activity diagram for our app “Toaster Roaster”.

### Use case diagram (system boundary)[Use Case Diagram](http://tinf11games.files.wordpress.com/2014/04/use-case-diagram.png)

### Activity diagram[Business Process Model](http://tinf11games.files.wordpress.com/2014/04/business-process-model.png)

# [Game Design Patterns](http://tinf11games.wordpress.com/2014/04/10/game-design-patterns/)

This post will describe the Game Design Patterns used in our game.

## Rule Patterns

Our game has a static rule set that does not change regardless of the level or unlockables used:

1. The user is shown a pattern he will have to redraw later on by the non-player character Fredo. The user decides when to start the game.
2. There is a toaster (called Roasty) that toasts the toast after the user toggled the switch. After a random amount of time, the toast jumps out.
3. While the toast jumps out and slowly falls on a plate, the user can redraw the pattern presented before on the toast (called Toasty). This process is also referred to as roasting.
4. The roasting process is rated based upon the accuracy oft he redrawn pattern.
5. However, the accuracy of the roasting can be altered by unlockables.

## Action Patterns

* [Gain Competence](http://gdp2.tii.se/index.php/Gain_Competence): As for all skill-based games, the player should have the possibility to get better over time. On the one hand the user gets better because he gets used to the device, the gameplay and the handling. On the other hand, the process of improvement is boosted by unlockables ([described here](http://tinf11games.wordpress.com/2014/03/21/unlockables/)). Unlockables influence game values such as the risk of burning the toast, the size of toast, the total time to roast the toast and the speed and precision of the roasting process.
* [Improved Abilities](http://gdp2.tii.se/index.php/Improved_Abilities): Unlockables help the user to improve his abilities by supporting the user: The Toast upgrades increase the toast’s size, whereas the Toaster upgrades increase the roasting time. Due to that, the player will be able to achieve higher scores.
* [Decreased Abilities](http://gdp2.tii.se/index.php/Decreased_Abilities): Toaster Roaster has also unlockables that reduce the abilities of the user. The roasting fire upgrades incorporate a higher risk of burning the toast. This effect increases with each upgrade. Due to this, the player can choose to play Toaster Roaster with a higher difficulty if the game gets to easy for the user.

## Event Patterns

* [Death Consequences](http://gdp2.tii.se/index.php/Death_Consequences): Whenever the user roasts a toast, there is a certain risk of damaging the toast by roasting it too long. If the toast is burnt, the player loses instantly, regardless of the accuracy of the pattern he has redrawn.

## Information Patterns

* [Loading hints](http://gdp2.tii.se/index.php/Loading_Hints): While the game is loading, there will be a visual clue denoting that the game won’t accept any user inputs during this interruption. However, during the loading process, the user will be shown some useful information as a benefit.
* [Easter Eggs](http://gdp2.tii.se/index.php/Easter_Eggs): For the amusement of the player, Toaster Roaster will incorporate a hidden Easter Egg, which will only be revealed if a player spends lots of time playing Toaster Roaster. Which Easter Egg will be implemeted, is a surprise.

## Goal Patterns

* [Goal Achievements](http://gdp2.tii.se/index.php/Goal_Achievements): Toaster Roaster will incorporate several achievements ([described here](http://tinf11games.wordpress.com/2014/03/19/achievements/)) that track goals the player has achieved while he played the game.

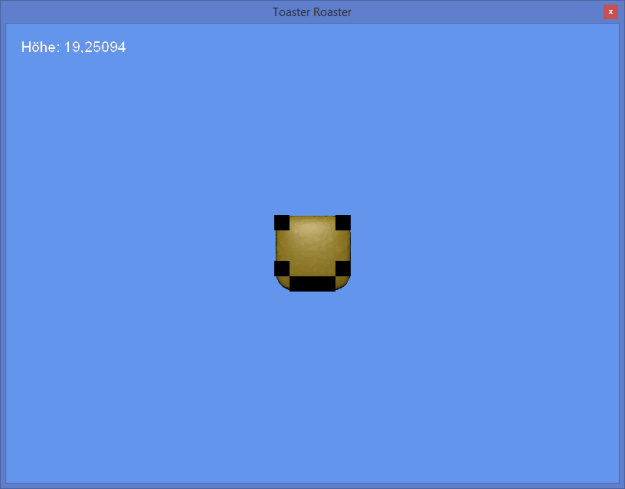
## Difficulty-Related Patterns

* [Casual Gameplay](http://gdp2.tii.se/index.php/Casual_Gameplay): Our game is a [casual game](http://en.wikipedia.org/wiki/Casual_game), which means that there has to be an easy onboarding process, a simple gameplay and short cycle times. In addition, it’s a [skill-based game](http://en.wikipedia.org/wiki/Online_skill-based_game), because you have to redraw a pattern which is determined by the game as best as possible.
* [Difficulty Levels](http://gdp2.tii.se/index.php/Difficulty_Levels): The difficulty of the roasting process is determined by the game values that are influenced by unlockables.

## Player Patterns

* [Single-Player Games](http://gdp2.tii.se/index.php/Single-Player_Games): Toaster Roaster is intended to be a single-player game without additional Non-Player Characters (NPCs). However, there will be some social components, such as a leaderboard.
* [Further Player Improvement Potential](http://gdp2.tii.se/index.php/Further_Player_Improvement_Potential): With the help of unlockables and by building skills, the user can get better over time.

**It works! Or: How we enabled roasting.**

* Good news: Last weekend our [team member](http://tinf11games.wordpress.com/2014/02/14/team-toaster-roaster-stellt-sich-vor/) Simon implemented the roasting process. This was a very important milestone, since the screenflow is now completed and the game works as described in [our concept](http://tinf11games.wordpress.com/2014/02/27/toaster-roaster-das-konzept/).
* [](http://tinf11games.files.wordpress.com/2014/04/roasting-works.png)
* Roasting in Toaster Roaster
* The cruical part showing how to enable roasting is shown in Simon’s [GitHub commit](https://github.com/ManuelRauber/ToasterRoaster/commit/6b86453b820fca5ceb7d1946aae4a3bae5733157).
* The trick was to overlay to graphics. The background texture represents the toast the foreground texture is generated by a boolean array. A click on the toast sets the corresponding value in the boolean array after the foreground texture is updated and the roast is shown.
* However, there are still some things to do: We will create some graphics in order to overhaul our design and will take a look at implementing Analytics using the [Localytics](http://localytics.com) service.

[**Toaster Roaster Highscore Services**](http://tinf11games.wordpress.com/2014/04/22/toaster-roaster-highscore-services/)

Today we implemented the highscore service of Toaster Roaster. It has two services for managing the highscore.

**Local highscore service**

The local highscore service saves the highscores of the user on the device the game is running. This is platform independent due to Wave Engines ability of a platform independent storage service.

The user does not have an option to disable to local highscore, since it will not be published anywhere.

**Azure highscore service**

The azure highscore service uses [Microsoft Azure](http://azure.microsoft.com/en-us/) as a backend storage provider. This highscore is available online and is used for a world wide competition of toast roasting.

The user has the option to disable a automatic highscore upload if he don’t want to enter the worldwide competition.

Unfortunately our subscription for Azure has expired, so we can’t use Azure anymore, but it is still implemented.

# Wave & Localytics (Signup and Basic Usage)

[1 Antwort](http://tinf11games.wordpress.com/2014/04/22/wave-localytics-signup-and-basic-usage/#comments)



Localytics logo

[Localytics](http://localytics.com) is the default analytics service for games built using the [Wave Engine](http://tinf11games.wordpress.com/2014/02/15/wave-engine/). However, the Wave Engine allows you to implement own analytics providers. We decided to use the default service and the existing implementation.

This blog post will describe the setup of the Localytics service and the basic usage implemented so far.

Localytics was founded in the year 2008 and focuses on mobile and web analytics and marketing. Currently, Localytics has more than 5,000 customers reaching 1.5 billion devices per month.

## Signup

Signing up at Localytics is quite easy: You just enter your basic data (name, e-mail, password), confirm your e-mail address and you are good to go. Localytics representatives are ready to assist you via Twitter or even phone.

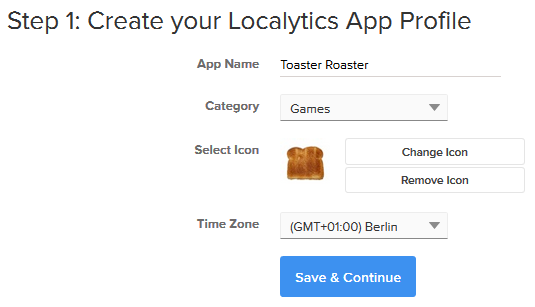
## Free trial – or isn’t it?

There is a free trial of Localytics and the provider claims that you do not have to enter credit card details. This is true; however, the standard version of Localytics is nearly useless, since you can’t add new applications (this requires the so-called new Localytics, which again requires a valid credit card). So, we decided to enter our credit card details and join Localytics as a regular customer.

The service is still free as Toaster Roaster won’t reach more than 25,000 monthly active users (MAUs) during its beta phase. If this limit would be exceeded (for example after a general release), the service is liable for costs, starting at $200 per month.

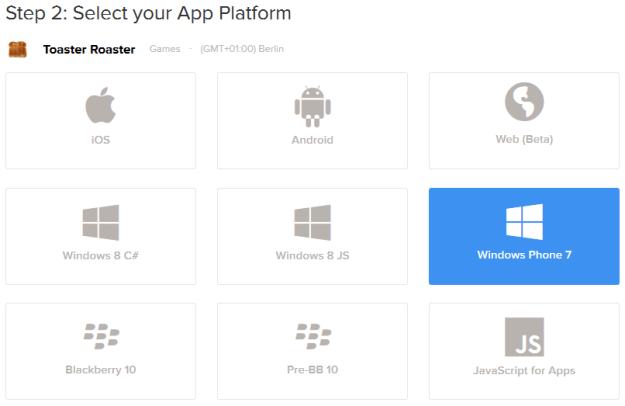
## Configuring Localytics

To use the analytics tools, you have to create an app profile first. This is quite easy: You just have to specify a name, a category (for comparisons with competitors), pick an icon for your app and finally select the correct time zone.

[](http://tinf11games.files.wordpress.com/2014/04/createapp.png)

Creating an app in Localytics

After that, you have to specify the correct platform for Localytics to work. Localytics provides several SDKs depending on the target platform you select in this step. The Localytics service of the Wave Engine always identifies itself as a Windows Phone application when talking to Localytics, regardless of the target operating system the game will be run on. Therefore, we picked Windows Phone 7 here.

[](http://tinf11games.files.wordpress.com/2014/04/createapp2.png)

Selecting the right target platform

## Configuring Wave

Configuring Localytics in the game project is basically as easy as:

\_localytics = new Localytics(application.Adapter, new LocalyticsInfo("19121dc67d0744823bcf572-3a407cd6-ca5c-11e3-9c53-009c5fda0a25"));

We introduced a new service called AnalyticsService which is registered in the Wave’s service manager. Therefore, the service can be obtained at any time while the game is opened.

The service is initialized during startup (i.e. a session is started) and offers two methods: TagEvent in order to create events with parameters and Close which is used to close a session.

You can also check the [responsible analytics service](https://github.com/ManuelRauber/ToasterRoaster/blob/master/Source/ToasterRoaster/ToasterRoaster.Game/Services/AnalyticsService.cs) in our GitHub repository. Please note that our real Localytics app ID is not publicly disclosed for the reasons we’ve noted above.

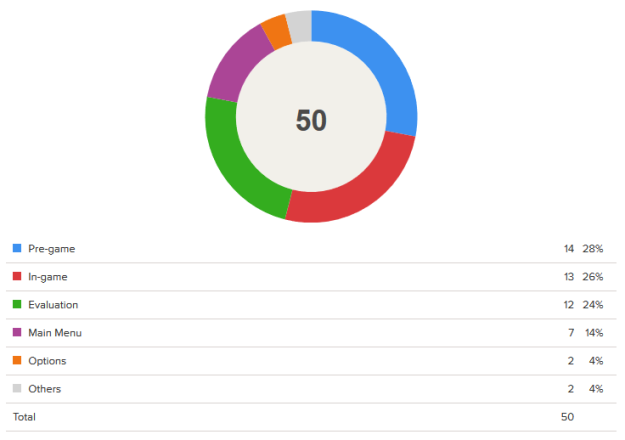
## Our first metrics

In our first draft, there are two metrics that we analyze so far:

### Pages opened

Whenever a new Game scene is drawn, we trigger the Analytics service and pass the page’s human-readable name as a parameter. Therefore, we are able to track how often certain pages are opened.

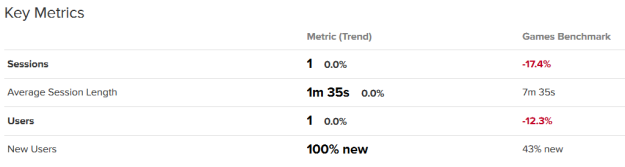
On the one hand, this helps identifying the heavily used parts of the game and on the other hand shows the pages that are less interesting for the user.

[](http://tinf11games.files.wordpress.com/2014/04/createapp3.png)

Pie chart showing the usage of the different pages

### Users vs. Sessions

Localytics seperates users from sessions. Whereas a user describes one and the same person, a session is a certain period of time spent by the user on using a game or visiting a website.

[](http://tinf11games.files.wordpress.com/2014/04/createapp4.png)

Overview: Users and sessions

A very important value is the average session length. This value tells us how long our game is open. On the right hand side, you can see the comparison with other apps from the Games category. In addition, Localytics tracks some additional information, e.g. the country the user resides in.

Our work for the next week is to evaluate our metrics and, if neccessary, add or remove some of them.

# [Local statistics](http://tinf11games.wordpress.com/2014/04/23/local-statistics/)

Today we implemented a Statistic Servcice to track some values while the player is playing the game.

It is a really simple service as shown in the mockup:

[](http://tinf11games.files.wordpress.com/2014/03/statistics.png)

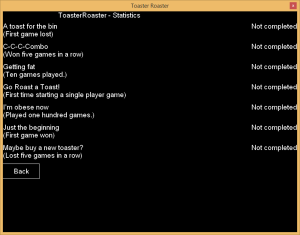
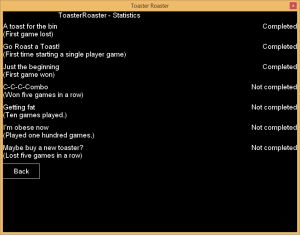
Currently we are tracking the following values:

* Total games played
* Overall accuracy
* Games won
* Games lost
* 3-Star games
* 2-Star games
* 1-Star games

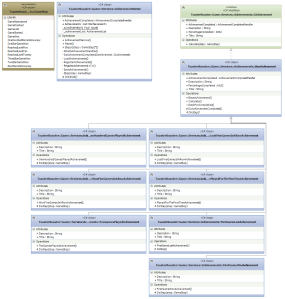
Within the options screen the user has the ability to reset the statistics. These statistics will not be published and stay at the user’s device.

# Achievements – Teil 1

Diese Woche haben wir Achievements implementiert, die dem Spieler lokal zur Verfügung stehen.

[](http://tinf11games.files.wordpress.com/2014/05/achievements1.png)[](http://tinf11games.files.wordpress.com/2014/05/achievements2.png)

## Implementierung

[](http://tinf11games.files.wordpress.com/2014/05/classdiagram.png)

Die Implementierung von Achievements gestaltet sich relativ einfach.

### AchievementService

Als applikationsweite Schnittstelle steht die Klasse “AchievementService” zur Verfügung. Sie enthält zum einen Logik, um die Achievements lokal zu speichern und zu laden. Zum anderen enthält sie Logik, um neue Achievements hinzuzufügen und den aktuellen Achievements den aktuellen Spielstand zu übermitteln. Es handelt sich hierbei um ein Observer-Pattern.

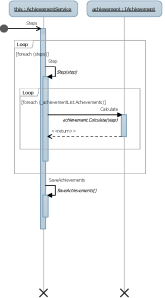
An verschiedenen Stellen im Code wird “AchievementService.Step(GameStep step)” aufgerufen. Bei “GameStep” handelt es sich um eine Enumeration, welche für Achievements interessante Werte enthält (z.B. Game won, Game lost, game started, reached level five, reached level ten). Der übergebene GameStep wird nun an alle registrierten Achievements weitergeleitet. Jedes Achievement entschiedet nun selbst, was es damit tut.

### BaseAchievement

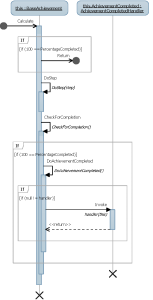
Diese Klasse implementiert das Interface “IAchievement” und stellt grundlegende Funktionen zur Verfügung. Sie implemetiert die Methode “Calculate” des Interfaces, um dort die abstrakte Methode “DoStep(GameStep step)” aufzurufen, welche von den einzelnen Achievements implementiert werden muss. Diese Methode enthält die Logik des Achievements.

Nach dem in “Calculate” die “DoStep()”-Methode aufgerufen wurde, wird die Methode “CheckForCompletion” aufgerufen, welche prüft, ob das Achievement nach dem Aufruf erhalten wurde.

Ein Sequenz-Diagramm soll den Ablauf deutlich machen:

[](http://tinf11games.files.wordpress.com/2014/05/sequencediagram1.png)

Aufruf AchievementService.Step

[](http://tinf11games.files.wordpress.com/2014/05/sequencediagram2.png)

Aufruf BaseAchievement.Calculate

Ein konkretes Achievement ist wie folgt [implementiert](https://github.com/ManuelRauber/ToasterRoaster/blob/master/Source/ToasterRoaster/ToasterRoaster.Game/Services/Achievements/WonFiveGamesInARowAchievement.cs):

[](http://tinf11games.files.wordpress.com/2014/05/achievementscode.png)

Implementierung eines Achievements

## Achievements

Folgende Achievements sind aktuell implementiert:

* A toast for the bin – First game lost
* Just the beginning – First game won
* Maybe buy a new toaster? – Lost five games in a row
* I’m obese now! – Played one hundred game
* Go roast a toast! – Played for the first time
* Getting fat – Ten games played
* C-C-C-Combo – Won five games in a row

Dank der einfachen Implementierung können weitere Achievements sehr schnell implementiert werden.

## Ausblick

Aktuell fehlt in dieser Variante noch die Implementierung des “AchievementCompletedHandlers”, der immer dann aufgerufen wird, sobald ein Achievement erfolgreich abgeschlossen wurde. In diesem Fall soll dem Spieler ein kleiner Hinweis angezeigt werden.