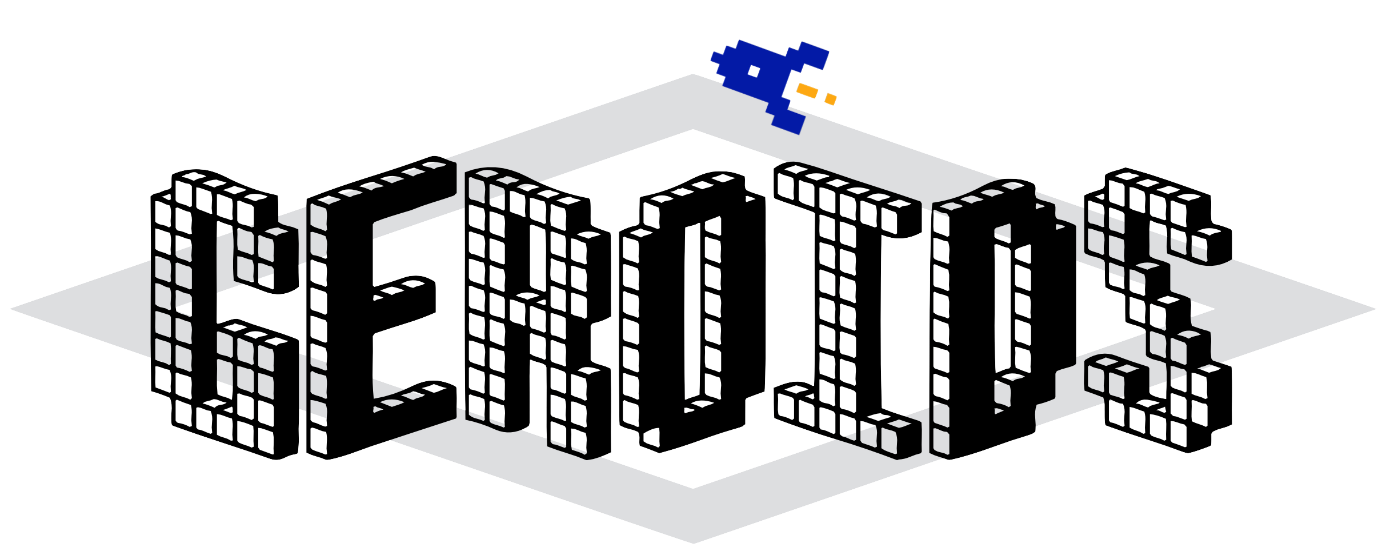
***Design Geroids***

Ein 2D-Arcadegame für Klein und Gross



Projektteam PSIT3 Gruppe 01

Arben Shabani (PL)

Linda Bödi

Valentin Bossi

Matthias Kaderli

**Inhaltsverzeichnis**

[**1.** **Projektmanagement** 3](#_Toc467517335)

[**1.1.** **Iteration 4 [Construction Phase]** 3](#_Toc467517336)

[**1.2.** **Iteration 5 [Construction Phase]** 4](#_Toc467517337)

[**1.3.** **Aktualisierte Risikoliste** 5](#_Toc467517338)

[**2.** **Architektur** 6](#_Toc467517339)

[**3.** **Design-Klassendiagram** 7](#_Toc467517340)

[**4.** **Klassenverantwortlichkeiten** 8](#_Toc467517341)

[**5.** **Zusammenarbeitsdiagramme** 11](#_Toc467517342)

[**6.** **Glossar** 13](#_Toc467517343)

[**7.** **GUI Design** 14](#_Toc467517344)

1. **Projektmanagement**

Die Hälfte der Construction Phase ist nun vorbei und die nächsten beiden Iterationsphasen sind geplant. Folgend ist noch einmal unsere grobe Phasenplanung aufgeführt, welche in der Elaboration-Phase leicht angepasst wurde.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Iteration | Start | Ende | Meilenstein | Aufwand [h] | Ziele / Umsetzung |
| Inception Phase | | | | | |
| 1 | 20.09.2016 | 04.10.2016 | M1 | 50 | Vision, Projektskizze, Präsentation |
| Elaboration Phase | | | | | |
| 2 | 04.10.2016 | 18.10.2016 |  | 60 | Entwicklungsumgebung, GUI-Entwurf |
| 3 | 18.10.2016 | 01.11.2016 | M2 | 80 | Domänenmodell, UML-Diagramm, DB-Entwurf, SW-Architektur |
| Construction Phase | | | | | |
| 4 | 01.11.2016 | 15.11.2016 |  | 100 | UC3 Implementierung |
| 5 | 15.11.2016 | 29.11.2016 | M3 | 90 | UC 1 & UC2 implementieren |
| 6 | 29.11.2016 | 06.12.2016 |  | 60 | UC4 & Datenbank Implementierung |
| Transition Phase | | | | | |
| 7 | 06.12.2016 | 20.12.2016 | M4 | 60 | UC1-4 GUI-Testing, Prototyp-Release |
| Total | | | | 500 |  |

* 1. **Iteration 4 [Construction Phase]**

Nachstehend ist die Iterationsplanung für die 4.Iteration und somit die erste Iteration der Construction Phase aufgeführt.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task # | Arbeitspaket | Aufwand [h] | Ist [h] | Verantwortlich |
| 1 | Statussitzung x2 | 8 | 8 | Alle |
| 2 | Implementierung Klassen Game, Account mit Tests | 20 | 10 | Arben S. (PL) |
| 3 | Implementierung Klassen Gamefield, Figure mit Tests | 20 | 10 | Linda B. |
| 4 | Implementierung Klassen Projectile, Geroid, Movement, Position mit Tests | 20 | 10 | Matthias K. |
| 5 | Implementierung Klassen Transmitter, Parser mit Tests | 20 | 10 | Valentin B. |
| 6 | JavaScript Frontend | 10 | 10 | Matthias K./ Valentin B. |
| 7 | Wissenstransfer | 2 | 4 | Alle |
| 8 | Zusätzliches Refactoring | / | 10 | Alle, wurde nicht eingeplant |
| Total |  | 100 | 72 |  |

* 1. **Iteration 5 [Construction Phase]**

Als letztes noch die Planung der jetzigen, fünften Iteration.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task # | Arbeitspaket | Aufwand [h] | Ist [h] | Verantwortlich |
| 1 | UC3 Tests | 20 |  | Alle |
| 2 | UC3 Refactoring Figure, Gamefield, Micro, | 2 | 2 | Matthias K. |
| 3 | UC3 Refactoring Account, Position | 2 | 2 | Valentin B. |
| 4 | UC3 Refactoring Game, Geroid, Projectile, Collisionhandler | 2 | 3 | Linda B. |
| 5 | UC3 Abbildung Objekte mit Bilder | 15 | 10 | Valentin B. |
| 6 | UC3 Score | 5 |  | Linda B. |
| 7 | UC3 Game Over Seite | 5 | 3 | Valentin B. |
| 8 | UC1 implementieren + Tests | 15 |  | Arben S. (PL) |
| 9 | UC2 implementieren + Tests | 10 |  | Linda B. |
| 10 | Wissenstransfer | 4 |  | Alle |
| 11 | Statusmeeting 2x | 8 |  | Alle |
| 12 | Javadoc aktualisieren | 2 | 2 | Alle |
| Total |  | 90 |  |  |

* 1. **Aktualisierte Risikoliste**

Nachfolgend ist die aktualisierte Risikoliste aufgeführt. Die Punkte 6 und 7 wurden nach unten korrigiert, da Wissen angeeignet wurde und die Punkte 3 und 5 sind neu hinzugefügt worden.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nr. | Name | Beschreibung | WSK | Schaden | Stufe | Priorität | Massnahme |
| 1 | Konkurrenz | In dieser Branche herrscht eine grosse Konkurrenz. Schlimmstenfalls geht das Spiel unter, was in kleiner Benutzerzahl resultiert. | 50% | Sehr hoch | H | 1 | Werbung, Mund-zu-Mund-Propaganda |
| 2 | Update | Es kann sein, dass das Programm nach einem Browser-Update nicht mehr läuft | 30% | Sehr hoch | H | 1 | Programmierung in Java und JavaScript, wobei beides noch sicher noch weitere Jahre vom Markt unterstützt wird (aufgrund der grossen Verbreitung) |
| 3 | Designüberlastung | Es verbinden sich zu viele Clients und unser Softwaredesign kann nicht damit umgehen. | 70% | Hoch | H | 2 | Es wird ein Stresstest gemacht, um ein Limit zu finden und dieses zu setzen, damit sich keine weiteren Clients verbinden. |
| 4 | Hacker-Attacken | Die Datenbank ist nicht gut gegen konzentrierte Hacker-Attacken geschützt, was eine Gefahr für die Nutzerdaten darstellen kann | 5% | Hoch | M | 3 | Möglicherweise einen Experten hinzuziehen. |
| 5 | Datenvolumen | Das Datenvolumen das an die Clients verschickt wird, ist zu gross, um ein ruckelfreies Spielen über das Internet zu ermöglichen. | 75% | Mittel | H | 1 | Die Struktur der einzelnen Datenpakete, die an den Client geschickt werden, sollen encodiert werden, um Datenvolumen zu sparen. |
| 6 | Spiele im Web | Es gibt kaum Wissen zu Spiel Implementierung im Web | 40% | Mittel | L | 3 | Möglicherweise muss zuerst noch etwas Wissen angeeignet werden, um das Spiel gut implementieren zu können. |
| 7 | Design | Es existiert kein grosses Knowhow bezüglich Game-Design | 30% | Klein | L | 3 | Da das Spiel sowieso minimalistisch gehalten wird, sollten dieses fehlende Knowhow keine allzu grosse Gefahr darstellen |

**Legende:** L: Low; M: Medium; H: High

1. **Architektur**

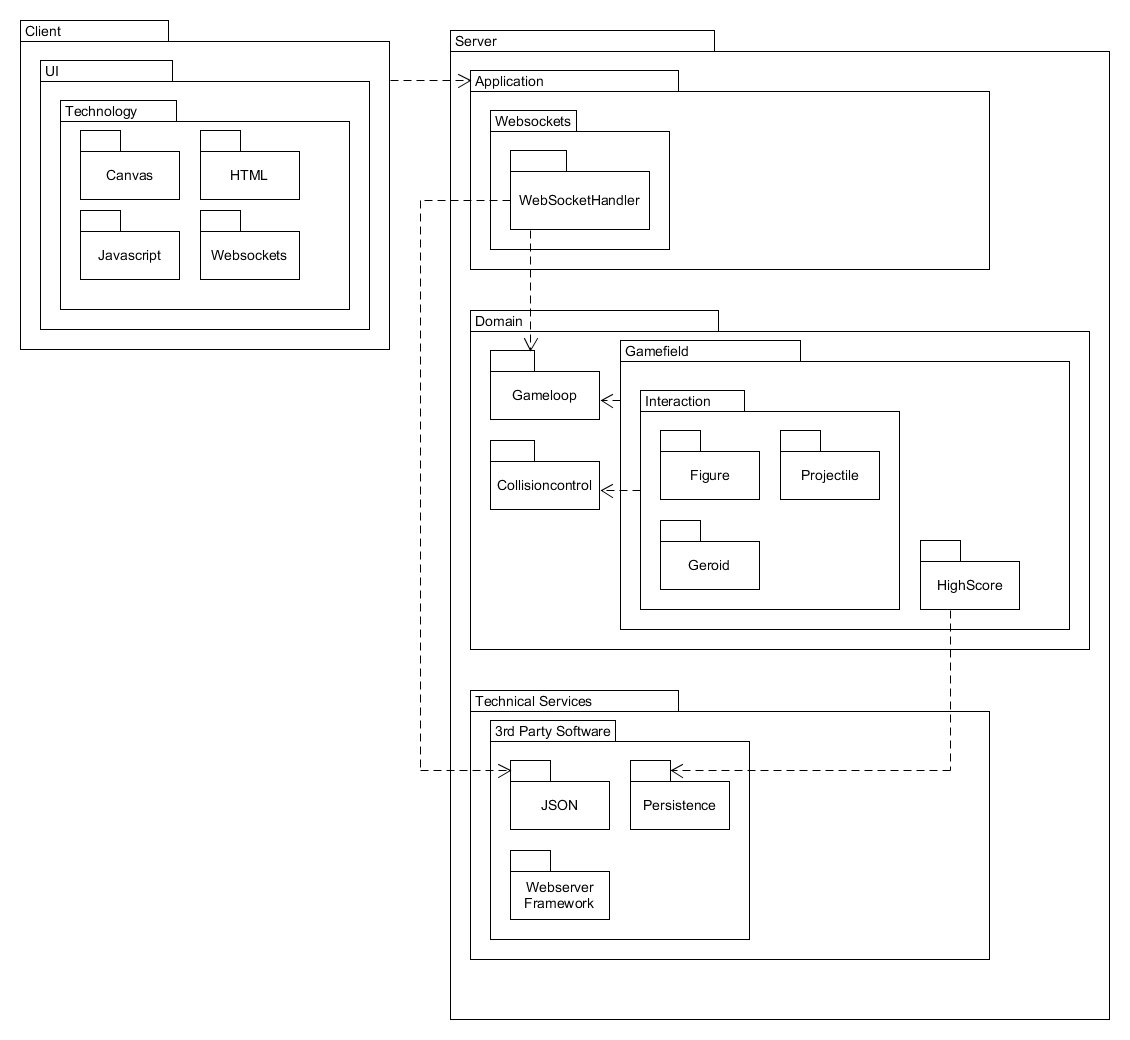
In Abbildung 1 ist die verwendete Architektur von *Geroids* ersichtlich. *Geroids* wurde als Web-App entwickelt und ist dementsprechend in einen Client- und einen Serverteil unterteilt. Die erweiterte Standartstruktur mit den Layern UI, Application, Domain und Technical Services wurde beibehalten. Das UI befindet sich auf der Clientseite; die restlichen drei Schichten Application, Domain und Technical Services befinden sich auf der Serverseite. Die Idee hinter dem Designentscheid ist, dass die Applikation möglichst Clientunabhängig laufen soll. Solange der Client Canvas unterstützt, ist der Client in der Lage das Spiel auszuführen. Für Jetty und für Websockets haben wir uns entschieden, weil wir erheblich von einer bidirektionalen und ständig offenen Verbindung zwischen Server und Client profitieren und mit Jetty mehrere Clients bedienen können.

Abbildung Architektur von Geroid

1. **Design-Klassendiagram**

**Design-Klassendiagramm/UML_Geroids_Klassendiagramm.pdf**In Abbildung 2 ist das Klassendiagramm von Geroids ersichtlich, mit dem Webserver als Fassaden-Controller.

Abbildung Design-Klassendiagramm von Geroids

1. **Klassenverantwortlichkeiten**

Folgend sind zu allen bestehenden Klassen in *Geroids*, die Klassenverantwortlichkeiten dargestellt.

|  |  |
| --- | --- |
| Klasse Account | |
| Die Klasse repräsentiert den Spieler mit seinen persönlichen Daten. | |
| Knowing | Name |
|  | Vorname |
|  | Nickname |
|  | E-Mail |
|  | Passwort |
|  | Skinliste |
|  | Typliste |
|  | Microtransactions |
| Doing | Erledigt das Lesen und Schreiben der Account-Daten |

|  |  |
| --- | --- |
| Klasse Figure | |
| Die Klasse stellt die Spielfigur, mit der der Spieler spielt mit ihren Ausmassen, ihrer Position und ihrem Bewegungsmustern zu Verfügung. | |
| Knowing | Die eigene Position und Ausdehnung |
|  | Das eigene Bewegungsmuster |
|  | Die eigene Form |
|  | Der eigene Typ |
| Doing | Eine Abbildung von sich selbst in einen JSON String oder ein JSONObject parsen |
|  | Die eigene Position um einen Schritt verändern anhand des Movements Objekts |

|  |  |
| --- | --- |
| Klasse Game | |
| Diese Klasse stellt den Fassaden Kontroller dar. Sie kontrolliert alle Spielfiguren auf dem Spielfeld und beendet das Spiel, falls es zu einer Kollision kommt. | |
| Knowing | Den Spieleraccount |
|  | Eine Liste mit allen "Geroids" |
|  | Die Punktzahl |
|  | Die Spielfigur des Spielers |
|  | Eine Liste mit allen Projektilen |
|  | Weiss ob es Kollisionen gibt |
|  | Kennt die Verbindungen mit dem Websocket |
|  | Wie viele "Geroids"maximal auf dem Spielfeld sein dürfen |
|  | Wann das letzte Mal geschossen wurde |
| Doing | Starten des Spiels |
|  | Aktualisieren des Spielfelds in regelmässigen Abständen |
|  | Erhöht die Positionen der Spielfiguren |
|  | Das Spiel abbrechen, falls die Spielfigur des Spielers mit einem "Geroid" kollidiert ist |
|  | Eine Abbildung des aktuellen Spielzustandes in ein JSONObject parsen |
|  | In zufälligen Abständen an zufälligen x-Positionen neue "Geroids"erzeugen |
|  | Eine Abbildung der aktuellen Liste der "Geroids" in einen JSON Array parsen |
|  | Eine Abbildung der aktuellen Liste der Projektile in einen JSON Array parsen |
|  | Eine neue Nachricht vom Websocket empfangen |
|  | Ein "Geroid" aus der Liste mit den "Geroids" entfernen |
|  | Ein Projektil aus der Liste mit den Projektilen entfernen |

|  |  |
| --- | --- |
| Klasse Gameserver | |
| Die Klasse stellt einen Server, der HTTP Requests beantwortet und Websocketverbindungen zulässt, zur Verfügung. | |
| Knowing | Verschiedene Handler, wie Websockethandler, HTTP Requesthandler und Kontexthandler |
|  | Speicherort statischer Dateien, wie HTML, Bilder usw. |
| Doing | Liefert statische Dateien, wie HTML, Bilder usw. auf HTTP Requests |
|  | Baut Websocketverbindung auf Wunsch des Clients auf |

|  |  |
| --- | --- |
| Klasse Geroids | |
| Die Klasse stellt eine geometrische Struktur mit ihren Ausmassen, ihrer Position und ihrem Bewegungsmustern zu Verfügung. | |
| Knowing | Die eigene Position und Ausdehnung |
|  | Das eigene Bewegungsmuster |
|  | Die eigene Form |
|  | Die eigene Identifikation |
| Doing | Die eigene Position um einen Schritt verändern anhand des Movements Objekts |

|  |  |
| --- | --- |
| Klasse Highscore | |
| Die Klasse stellt die Funktionalität zur Verfügung, um Abfragen an die Datenbank zu tätigen. | |
| Knowing | Score |
|  | Spieleraccount |
|  | Highscoreliste |
| Doing | Fragt die Datenbank nach Scores ab |
|  | Persistiert Scores in die Datenbank |

|  |  |
| --- | --- |
| Klasse Microtransactions | |
| Die Klasse stellt die Funktionalität zur Verfügung, um mit externem Zahlungssystem zu kommunizieren. | |
| Knowing | ID des Spielers |
|  | Spieleraccount |
|  | Microtransactionstype |
| Doing | Baut Verbindung zu externem Zahlunsgssystem her |
|  | Speichert und lädt Microtransactions des Accounts in die Datenbank |

|  |  |
| --- | --- |
| Klasse Movement | |
| Die Klasse stellt ein 2D-Bewegungsmuster für ein Objekt zu Verfügung, welches sich für jeden Tick selbständig bewegen soll. | |
| Knowing | Die eigene Geschwindigkeit in x- und y-Richtung |
| Doing | Erhöhung der eigenen Geschwindigkeit in x- oder y-Richtung um einen bestimmten Faktor |

|  |  |
| --- | --- |
| Klasse MyWebsocketHandler | |
| Die Klasse stellt die erforderlichen Websocket API zur Verfügung und ermöglicht das Erstellen einer neuen Spielinstanz. | |
| Knowing | Die Websocket Session zum Client |
|  | Die Spielinstanz |
| Doing | Erledigt den Datenaustausch zwischen der Spieleinstanz und dem Client |
|  | Erstellt ein neues Spiel bei einem erfolgreichen Websocket-verbindungsaufbau |

|  |  |
| --- | --- |
| Klasse Playscore | |
| Die Klasse stellt die Berechnung der Punktzahl zur Verfügung. | |
| Knowing | Seine aktuelle Punktzahl |
|  | Anzahl Punkte, wenn ein "Geroid" mit einem Projektil getroffen wird |
| Doing | Die Punktzahl erhöhen, falls ein "Geroid" mit einem Projektil getroffen wird |

|  |  |
| --- | --- |
| Klasse Position | |
| Die Klasse stellt eine Angabe zur Position, zusammen mit den Längen in x- und y-Richtung zur Verfügung. | |
| Knowing | Die eigene x- und y-Koordinate |
|  | Die eigene x- und y-Ausdehnung |
|  | Eine Liste mit Positionen für jeden Punkt für komplexe Formen |
| Doing | Die eigene Position aufgrund eines übergebenen Bewegungsmusters um einen Tick anpassen |
|  | Eine Abbildung von sich selbst in ein JSONObject parsen |

|  |  |
| --- | --- |
| Klasse Projectile | |
| Die Klasse stellt die Funktionalität für die Erstellung eines Projektils, seine Position und sein Bewegungsmuster zu Verfügung. | |
| Knowing | Das Spiel, in dem sich das Projektil befindet |
|  | Die eigene Position und seine Ausmasse |
|  | Das eigene Bewegungsmuster |
| Doing | Die eigene Position aufgrund eines übergebenen Bewegungsmusters um einen Tick anpassen |
|  | Eine Abbildung von sich selbst in ein JSONObject parsen |

|  |  |
| --- | --- |
| Klasse Skin | |
| Die Klasse stellt das Aussehen der Spielfigur, mit der der Spieler spielt, zur Verfügung. | |
| Knowing | Sein Name |
|  | Ein Bild der neuen Spielfigur |
| Doing | - |

|  |  |
| --- | --- |
| Klasse Type | |
| Die Klasse stellt den Typ der Spielfigur, mit der der Spieler spielt, mit ihrem Bewegungsmustern und den Besonderheiten zur Verfügung. | |
| Knowing | Sein Name |
|  | Die eigene Bewegung |
|  | Der Lebensbonus |
| Doing | - |

1. **Zusammenarbeitsdiagramme**

In Abbildung 3 ist das Kommunikationsdiagramm für die Systemoperation checkCollision() des UC3 Spielen ersichtlich. In Abbildung 4 ist das Sequenzdiagramm für die Systemoperation updateGamefield() ersichtlich.

Zusammenarbeitsdiagramme/Kommunikationsdiagramm_checkCollision.pdf

Abbildung Kommunikationsdiagramm UC3 Spielen - checkCollision

Zusammenarbeitsdiagramme/Sequenzdiagramm_updateGamefield.pdf

Abbildung Sequenzdiagramm UC3 Spielen - updateGamefield

1. **Glossar**

***Browser:*** Computerprogramm zur Darstellung von Webseiten im World Wide Web

***Geroids:*** Arcadespiel als Webapplikation; Name aus der Verschmelzung der beiden Wörter Geometry und Asteroids, eines der ersten Arcadespiele. Die Spielapplikation G*eroids w*ird immer kursiv geschrieben, um es von den Spielgegnern, den "Geroids", zu unterscheiden

***"Geroids":*** Spielgegner; geometrische Form, die durch das All fliegen, immer in Anführungs- und Schlusszeichen geschrieben, um es von der *Geroids-*Applikation zu unterscheiden

***Highscores:*** Die höchsten Punktstände, die erreicht wurden

***HTTPS:*** HyperText Transfer Protocol Secure (sicheres Hypertext-Übertragungsprotokoll) ist ein sicheres Kommunikationsprotokoll im World Wide Web

***In-App-Kauf:*** Käufe, die man innerhalb einer Applikation tätigen kann

***Instanz:*** Ein konkretes Objekt (in der Informatik)

***Nickname:*** Spitzname

***OO-Software-Design:*** Objektorientiere Software Programmierung

***Score:*** Punktestand

***JSON:*** Datenformat zum Datenaustausch zwischen Anwendungen

***Canvas:*** HTML Element für Zeichnungsflächen

***Jetty:*** Webserver

***Websocket:*** Internetprotokoll für bidirektionale Verbindungen

1. **GUI Design**

Folgend wird der aktuelle Stand des GUI’s gezeigt. Innerhalb des GUI’s handelt es sich um einen geradlinigen Ablauf. Das heisst, das GUI besitzt im Moment noch keine Verzweigungen, diese werden darum auch nicht gezeigt. Es gibt nur einen Weg zum Spiel:

1. Über die Startseite zur
2. Auswahlseite zur
3. Spieleseite.

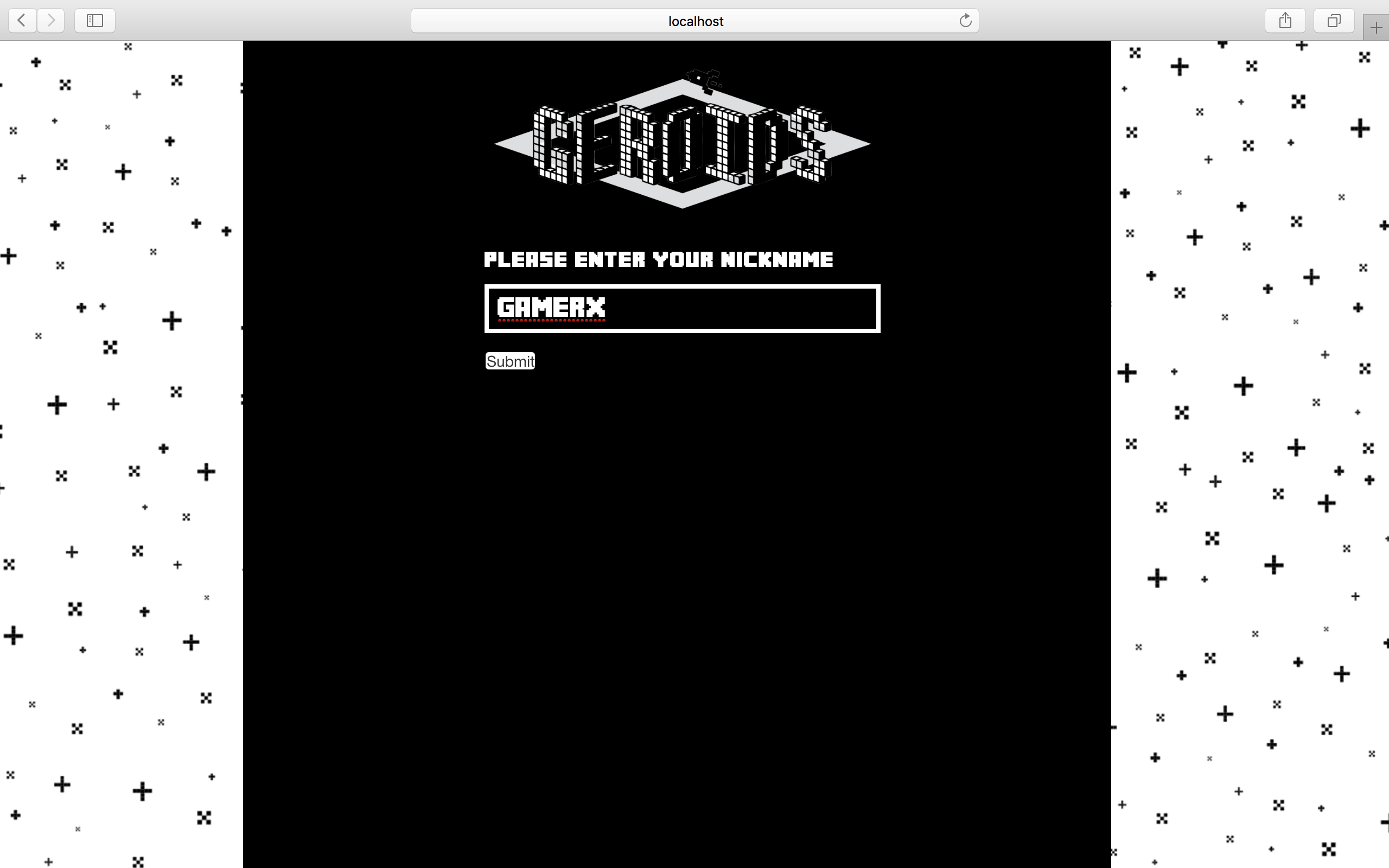
   
Auf Abbildung 5 sieht man die genannte Startseite. Dort wählt der Spieler seinen Nicknamen aus, den er für sein Spiel haben möchte.

Abbildung GUI Startseite

Die Abbildung 6 zeigt die einzige Folgeseite der Startseite, die Auswahlseite. Hier kann der Spieler ein neues Spiel starten, die vergangenen Scores oder das Impressum ansehen.

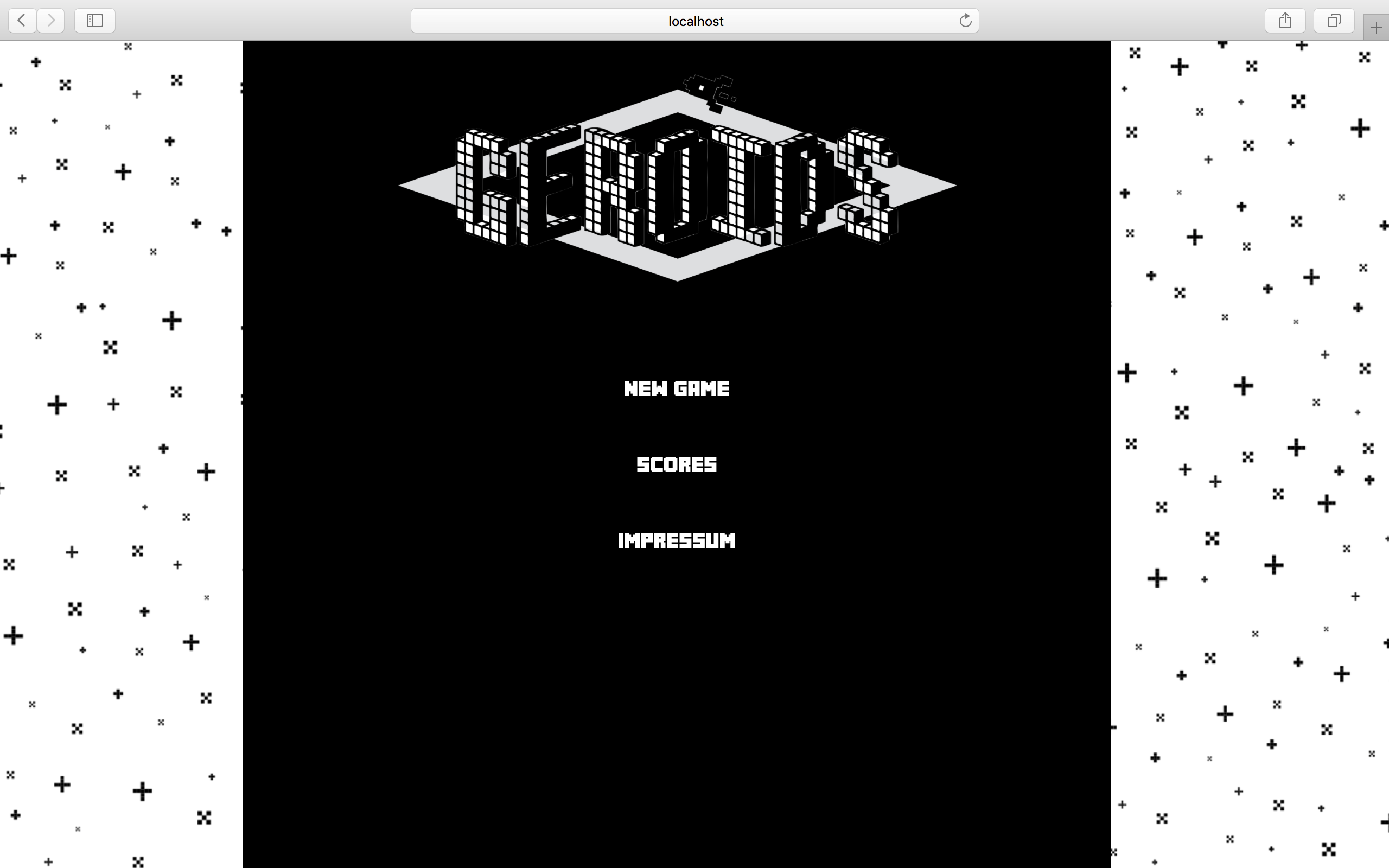


Abbildung GUI Auswahlseite

Die Abbildung 7 zeigt die Spieleseite. Auf dieser Seite spielt der Spieler das Spiel.

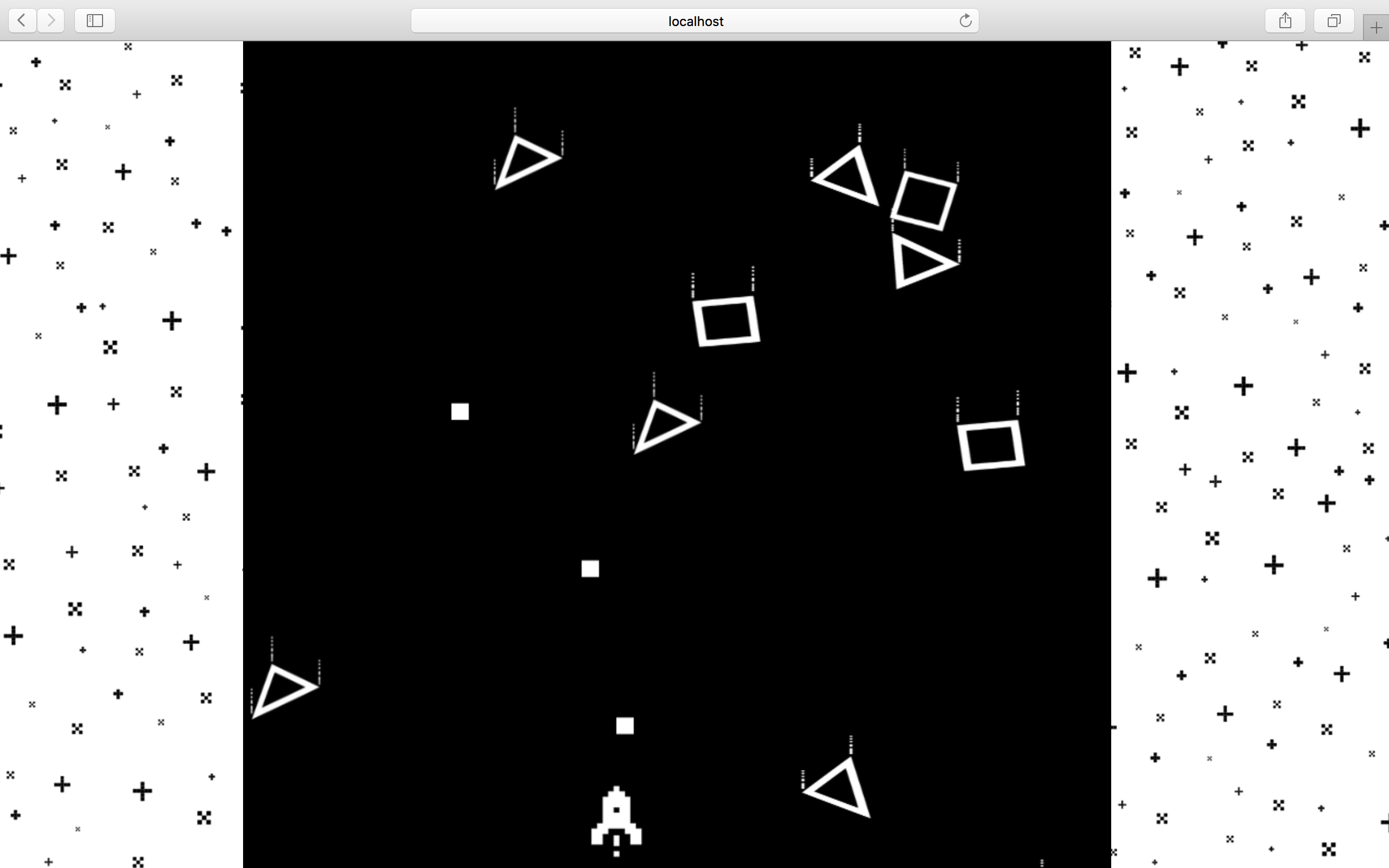


Abbildung GUI Spielseite

Abbildung 8 zeigt die Oberfläche, wenn ein Spiel vorbei ist.



Abbildung GUI Game Over Seite

Abbildung 9 zeigt, dass sich der Inhalt des GUI’s automatisch der Bildschirmgrösse anpasst und das Spiel somit auf den unterschiedlichsten Geräten spielebar wird.

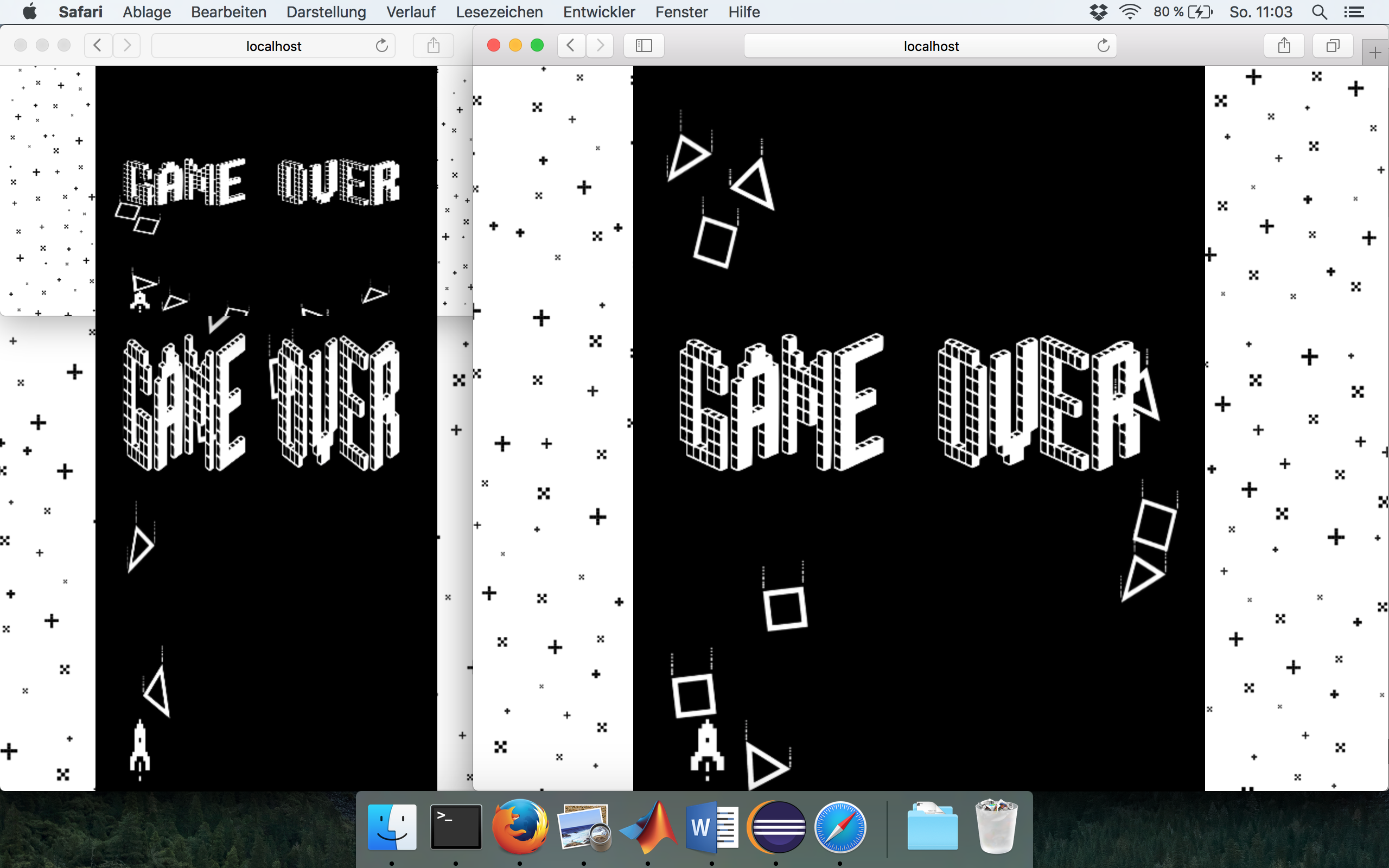


Abbildung GUI responsive Design

Abbildung 10 ist eine Bildschirmaufnahme von einem iPad, was zeigt, dass das GUI gut skaliert.

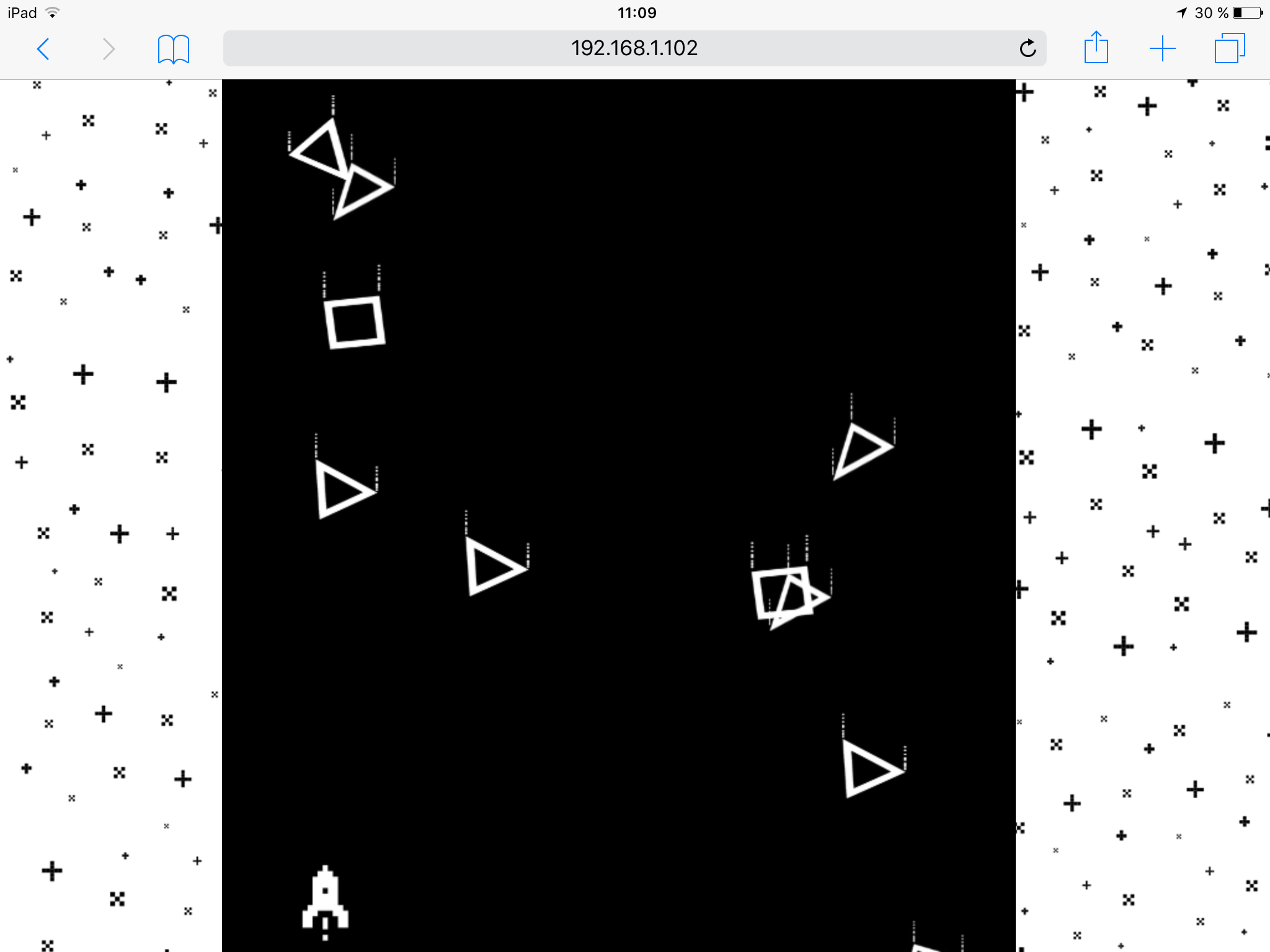


Abbildung GUI Test auf iPad