2016

Fruit vs. Veggies

Steffen Buhl ,Georg Fischer, Benjamin Strauß, Kai Plautz

IT14 A

30.6.2016

Tagebuch

- **12.05.2016**
  -Projektvorstellung von Herr Jenet.
  -Nachdem uns die Projektanforderungen erklärt wurden diskutierten wir und entschieden uns für ein grobes Spieleprinzip.
  - Über verschiedene Spieldesign Entscheidungen diskutiert und abgestimmt und diese Ideen in einem Dokument gesammelt.

- **19.05.2016**
  -Es wurde besprochen welche Klassen benötigt werden und deren Attribute. Außerdem wurde schon grob eingeteilt wer sich um welche kümmern wird.
  -Jeder hat sich Git und TortoiseGit auf seinen Laptop installiert und Benjamin hat jeden kurz geklärt wie es zu bedienen ist.
  -Die weitere Programmierlogik wurde besprochen

- **26.06.2016**
  -Es wurde das Projekt angelegt und das Monogame Framework eingepflegt
  -Die ersten Tileklassen und Ableitungen wurden angelegt und geschrieben
  -Die weitere Struktur des Programmes wurde besprochen

- **02.06.2016**
  -Es wurde nochmal über die Units diskutiert und auch die erste Unitklasse Ableitungen angelegt und geschrieben.
  -Der XMLLoader wurde besprochen und angefangen zu programmieren
  -Die ersten Grafiken wurden implementiert
  -Die erste Testmap wurde generiert aber die Instanziierung wurde nicht bedacht.

- **09.06.2016**
  -Der ursprüngliche Plan den Map Editor als eigenständiges Projekt zu programmieren erwies sich als unnötig kompliziert und somit wurde begonnen ihn ins richtige Projekt zu implementieren.
  -Die ersten Menüs wurden programmiert.
  -Die AttackUnit Klasse wurde erweitert und der FightManager wurde angelegt und angefangen zu programmieren.
  16.06.2016
  - Sound Unterstützt wurde implementiert
  - FightManager & AttackUnit wurde fertig programmiert. XMLLoader um das Laden von Units erweitert.
  -FactoryTile angelegt und wurde angefangen zu programmieren.

- **23.06.2016**
  -An den Schul-PC Kompatibilitätsprobleme mit dem Spiel entdeckt und versucht zu beheben. Leider ohne Erfolg
  -Projekt um CaptureTile und BaseTile erweitert und programmiert.

Inhaltsverzeichnis

3.1 Fruit vs. Veggies

Fruit vs. Veggies wird ein kleines Rundenbasierendes Strategiespiel. Ein Spieler übernimmt die Fraktion des Gemüse der andere die Fraktion des Obsts. Jeder Spieler hat eine Basis und Ressourcen. Es gibt verschiedene Felder auf den man entweder gegen Ressourcen neue Einheiten rekrutiert oder solange man sie besetzt Ressourcen einnimmt. Man kann die Gegnerischen Einheiten vernichten in dem man zu ihnen vorrückt und attackiert. Gewonnen hat der Spieler der zuerst die gegnerische Basis zerstört.

3.2 Beschreibung der Projektarbeit

Die Teilaufgaben

- Kacheln und Units müssen auf dem Bildschirm angezeigt werden.
- Die Map muss angezeigt werden.
- Wir müssen Einheiten auf der Map anzeigen können.
- Der Mapeditor muss geschrieben werden
- Sound muss in das Projekt eingebracht werden und zum Mapeditor hinzugefügt werden
- Einheiten müssen sich auf der Karte bewegen können.
- Ressourcen müssen auf die einzelnen Spieler verteilt werden.
- Es muss eine Klasse geschrieben werden die sich um den Ablauf der Kämpfe und Eroberungen kümmert.

  - Ressourcen werden als Währung benötigt um neue Einheiten zu rekrutieren
  - Kämpfe werden später von einer Klasse gemanaged und ist dafür notwendig die Spieler gegenseitig ihre Einheiten vernichten können um die Oberhand zu gewinnen.
  - Eroberungen sind wichtig um später an neue Ressourcen zu kommen. Es wird spezielle Gebäude geben die man einnehmen kann um jede Runde einen festen Beitrag an Ressourcen zu bekommen um somit neue Einheiten zu finanzieren. Auch kann man externe Rekrutierungsgebäude erobern um somit neue Einheiten direkt dort ins Spiel zu rufen wo man sie benötigt. Um das Spiel zu gewinnen wird man auch die gegnerische Basis einnehmen müssen.
  - Soundeffekte tragen zur Atmosphäre bei und werten das Projekt rundum auf.
  - Die Karte ist in Kacheln aufgeteilt und es gibt auch verschiedene Typen wie Hauptbasen, Rekrutierungskacheln, Ressourcenkacheln oder auch Umgebungskacheln die nicht durchgehbar sind wie Berge etc.
  - Der Map Editor wird benötigt um dynamisch und schnell neue Karten anfertigen zu können.
  - Es wird verschiedene Einheiten geben mit unterschiedlichen Kosten, Lebenspunkten und Angriffswerten um verschiedene Strategien zu ermöglichen.
  - Aufgaben wie Bewegung, Grafiken anzeigen usw. gehören zur einfachen Spiellogik und werden benötigt das das Spiel überhaupt lauffähig ist.

Technische, Organisatorische und Zeitliche Vorgaben

Die Zeitliche Vorgabe war gegeben und war von dem 12.05.2016 bis zum 30.06.2016. Wobei hier die Donnerstage in der Schule nicht ausreichen so musste jeder außerschulisch noch einiges tun.
Für die Technischen Vorgaben brauchte jeder einen Laptop den er zu jedem gemeinsamen Termin auch dabei haben musste. Des Weiteren benötigten wir Internet um unser Projekt auf einen gemeinsamen Stand zu bringen. Hierzu benutzten wir das Ressourcenmanagement System Git mit der Erweiterung TortoiseGit.

Als Framework haben wir das Framework Monogame benutzt.
Um uns wiederrum besser organisieren zu können benutzten wir Skype, WhatsApp

und auch wie
schon angesprochen Git. Diese begleitete uns das ganze Projekt über.
Weitere Vorgaben haben wir aus dem gegebenen Arbeitsblatt entnommen.

3.3 Ressourcen-und Ablaufplanung

An dem Projekt arbeiteten folgende Personen: Steffen Buhl, Georg Fischer, Benjamin Strauß sowie Kai Plautz. In den folgenden Tabellen können wir folgendes entnehmen: Wer welche Aufgaben erledigt hat und wie viel Zeit hierfür benötigt wurde.

Benjamin Strauß	Aufgaben	Stunden
	Projektanlegen und Framework pflegen	2
	Graphics Objekt angelegt und Getestet	1
	Teilklasse Angelegt und diese Getestet	1
	Resourcenteil Abgeleitet(Angelegt)	2
	Map Klasse erstellt	2
	Projekt auf windows formes Interpretiert (Kai und Benjamin)	4
	Sound Unterstützung	2
	Verbergen des Map Editors sowie anzeigen	1
	Menüs erstellt	3
	Cursor	2
	Mergen	

Kai	Aufgaben	Stunden
	XML Loader	4
	Teilklasse	1
	Menüklasse	3
	Map Editor	6
	FactoryTile	2
	Cursor	1

Steffen Buhl	Aufgaben	Stunden
	Unitklasse erweitert	1
	AttackUnit geschrieben	3
	FightManager geschrieben	2
	XML Loader um das laden von Units erweitert	2
	CaptureTile & BaseTile geschrieben	2
	Doku schreiben	7

Georg Fischer	Aufgaben	Stunden
	Playerklasse angelegt	**1**
	Grafiken erstellen	**12**
	Menüklasse	**1**
	Doku schreiben	**9**

Die Gesamtstunden Beliefen sich so auf Insgesamt: **77 Stunden**.

Die Reihenfolge

1. Erstellen der Grafiken
2. Projektanlegen und Framework pflegen
3. Graphics Objekt angelegt und Getestet
4. Teilklasse
5. XML Loader
6. Units, Ableitung der Teilklasse
7. Map Klasse
8. Abgeleitete Unit Klassen, FightManager geschrieben
9. Map Editor
10. Menü-, Windows-, Paintklasse
11. Soundklasse
12. Cursorklasse

3.4 Tätigkeitsangaben mit Zeitraster

Was wurde wann und vom wem in welcher Phase durchgeführt

Wo konnte Zeit eingespart werden gegenüber der Zeitplanung

Leider konnten wir in unserer Zeitplanung keine Zeit einsparen.

Wo wurde mehr zeitbenötigt

Es wurde viel Zeit benötigt um das Spiel auf den Schul-PCs zum Laufen zu bringen. Durch das Monogame Framework wollte das Spiel nicht auf den Schul-PCs funktionieren und es wurden viele Stunden in der Schule damit verschwendet diesen Fehler zu beheben, leider ohne Erfolg. Somit wurde auch Zeit verschwendet um das Monogame Framework wieder aus dem Projekt zu entfernen. Viel Zeit ging auch verloren, weil kein gescheiter Zeitplan anfangs erstellt worden ist und es am Ende des Projekts viel Zeitdruck gab und sehr viel noch zu Hause erledigt werden musste. Auch das manchmal Klassen und Funktionen hinzugefügt wurden ohne zu testen ob sie funktionieren kostete am Ende etwas Zeit. Auch die Integration des Map Editors in die Solution dauert sehr lange.

3.5 Durchführung und Auftragsbearbeitung

Vorgehensweise

In der Schule wurde immer zuerst besprochen wie weit jeder gekommen ist und ob es Probleme irgendwo gab. Danach wurde besprochen wer welche Aufgaben zusammen mit wem(Immer in Zweiergruppen) übernimmt und was das Tagesziel sein sollte. Am Ende der Unterrichtsstunde wurde überprüft ob das Tagesziel erreicht wurde und besprochen wer was bis zum nächsten Donnerstag zu erledigen hat.Nachdem Unterricht pushte auch jeder über TortoiseGit seine Fortschritte den er in Schule machte so dass jeder den aktuellsten Stand des Projekt hat. Zuhause gab es Absprachen per WhatsApp oder Skype Konferenzen. Hier wurden Probleme besprochen, Spieldesign Entscheidungen getroffen und geklärt was noch benötigt wird.

Entscheidungen getroffen

Durch Diskussionen in der Schule wie auch per Skype wurden gemeinsame Entscheidungen getroffen. Nur selten wurde etwas nicht gemeinsam entschieden. Bei den Diskussionen hat Herr Benjamin Strauß uns immer begleitet und durch seine Erfahrung in anderen Außerschulischen Projekten gute Ratschläge geben können die meist sehr hilfreich waren.

Abweichungen von Teilzielen

Leider wurde zu spät getestet ob das Monogame Framework lauffähig auf den Schul-PCs ist und führte somit zu Abweichungen und Verschiebungen im Zeitplan. Das Monogame Framework musste entfernt werden um das Spiel auf den Schul-PCs zum Laufen zu bringen. Hierbei half dass der Map Editor bereits auf Windows Forms lief.

3.6 Qualitätssicherung

Unser Programm wurde von allen Mitgliedern der Gruppe getestet. Dies geschah in dem wir ein normales Spiel einmal simuliert haben. Fehler die wir gefunden haben wurden in Skype besprochen und daraufhin wieder untereinander aufgeteilt und behoben. Nach dem erfolgreichen behebenden einzelnen Fehler wurde das Spiel erneut von jedem Durchgespielt dies wurde so lange wiederholt bis wir keine Fehler mehr fanden. Die Ziele die wir uns am Anfang des Projektes festgelegt haben wurden erreicht nach und danach wurde noch an Feinheiten gearbeitet.Das Spiel wurde extra so programmiert das es leicht zu erweitern und verbessern ist. Somit lässt sich das Spiel sehr leicht um weitere Einheiten, Gebäude und weitere Extras erweitern. Ein Ziel das wir nicht vollends zu unserer Zufriedenheit erreicht haben waren die Grafiken. Leider beanspruchten diese sehr viel Zeit, weil sie alle per Hand gefertigt wurden. Somit sind nicht alle Grafiken perfekt, weil letztendlich die Fertigstellung des Projektes unsere Priorität war.

3.7 Reflexion der Vorgehensweise im des Ergebnisse

Was war Gut?

Es war eine gute Idee Git zu nutzen, somit konnten wir alle unabhängig an dem Projekt weiterarbeiten, aber in Regelmäßigen Abständen unseren Code mergen und jeder war auf dem neusten Stand des Projekts ohne mühseliges zusammensetzen und hin und her schicken des neuen Codes. Auch das wir von Anfang an darauf geachtet haben alles so Objektorientiert wie möglich aufzubauen war eine gute Entscheidung. So konnten wir ohne Probleme immer das Projekt um weitere Dinge erweitern, weil alles dynamisch ist und so gut wie nichts statisch ein programmiert wurde. Auch die Gruppenkommunikation außerhalb der Schule war sehr gut. Durch regelmäßige Absprachen per WhatsApp oder stundenlangen Konferenzen in Skype war jeder immer auf dem neusten Stand und alle Entscheidungen wurden auch immer zusammen getroffen.

Was hat sich im Nachhinein als schlecht gezeigt?

Das Monogame Framework war im Nachhinein ein Fehler, weil wir nicht direkt getestet haben ob das Programm damit auf den Schul-PCs läuft, somit wurden einige Stunden mit der Integration des Frameworks und mit dem Entfernen verschwendet. Auch das wir keine feste Termine für „Meetings“ festgelegt haben erschwerte es manchmal Konferenzen in Skype zusammenzurufen. Auch das wir keinen gut strukturierten und überlegten Zeitplan hatten führte zu zahlreichen Problemen am Ende, so dass wir am Ende Zeitdruck hatten. Auch das nicht immer alles getestet wurde nachdem es programmiert wurde und dann später zu Fehler führte hat einige Zeit und Nerven gekostet,

Was würden wir in Zukunft anders machen?

Das zuerst ausgiebig getestet wird bevor es in Git gepusht wird. Das bezieht sich auf den Fehlerhaften Code der ein das Projekt eingefügt wurde ohne zu testen und auch auf das zu späte Testen ob das Monogame Framework auf den Schul-PCs funktioniert.

In Zukunft würden wir auch einen festen und strukturierten Zeitplan erstellen für verschiedene Meilensteine und auch Konferenzen außerhalb der Schule. So hätte man sich viel Zeit und Stress am Ende erspart.

Vergleichen der Zeit mit der tatsächlichen Zeit

Leider wurde keine detaillierteren strukturierteren Zeitpläne für das komplette Projekt erstellt, sondern nur für die jeweiligen nächsten Schultermine den wir zwar eingehalten haben, aber nicht weit genug durchdacht war. Wie bereits thematisiert war der ein Fehler und würde in der Zukunft nicht wiederholt werden. Der allg. vorgegebene Zeitraum wurde aber eingehalten.

3.8 Anhang

Anleitung:

Fruit vs. Veggies ist ein kleines Rundenbasierendes Strategiespiel. Ein Spieler übernimmt die Fraktion des Gemüse, der andere die Fraktion des Obsts. Jeder Spieler hat eine Basis und durch Einnehmen verschiedener Ressourcenfelder und rekrutieren neuer Einheiten geht es darum den gegnerischen Spieler zu besiegen, indem man seine Basis erobert.

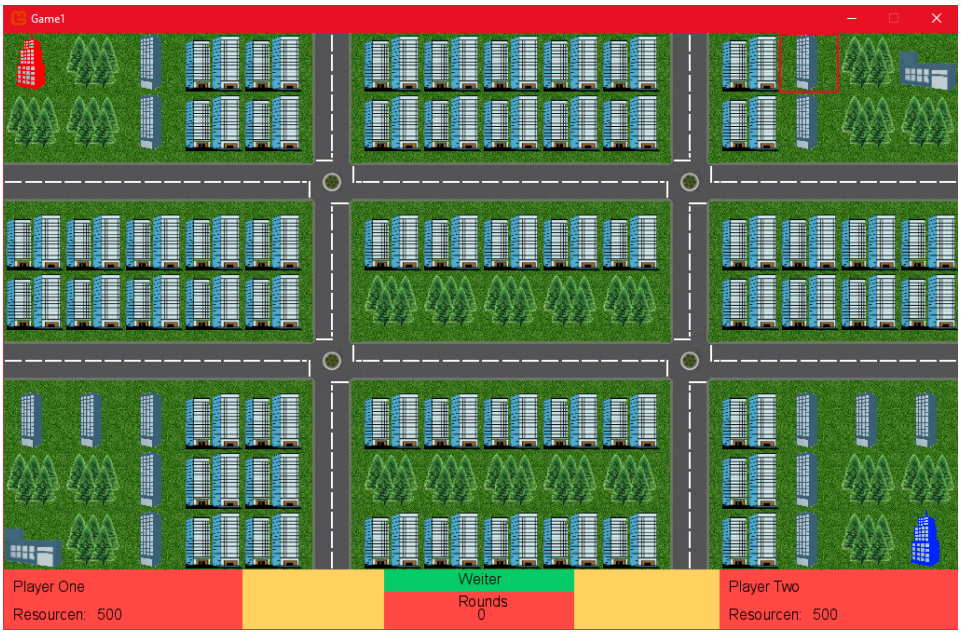

ABB1: Ansicht der Startmap mit Statusleiste am unteren Bildrand.

Auf der linken Seite sind die Ressourcen von Spieler 1 zu sehen, auf der rechten Seite die von Spieler 2. In der Mitte kann man sehen, in welcher Runde sich das Spiel momentan befindet. Durch Bestätigen des grünen „Weiter“-Knopfs beendet jeder Spieler seine Runde und der andere Spieler ist dran.

Am Anfang des Spiels hat jeder Spieler nur eine Hauptbasis. Die Hauptbasis von Spieler 1 ist rot (zu sehen oben links) die von Spieler 2 dunkelblau (rechts unten). Um zu gewinnen muss man die Basis des jeweiligen anderen Spielers erobern.

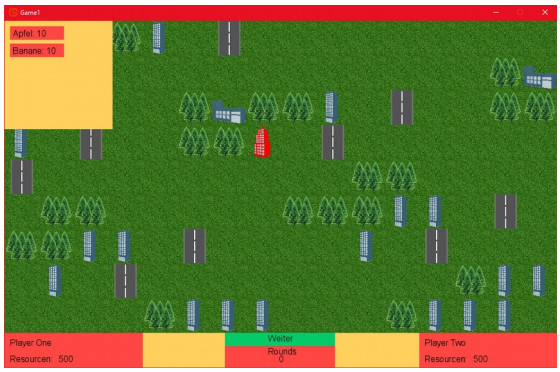

ABB2: Hier ist das Kaufmenü zu sehen. Dieses öffnet sich, wenn ein Spieler während seiner Runde eine von ihm kontrollierte Fabrik oder seine Hauptbasis anklickt.

Er kann eine Einheit aus diesem anklicken und die entsprechende Einheit erscheint auf der Fabrik oder Basis. Natürlich kostet das Erstellen von Einheiten Ressourcen, reichen diese nicht, so können keine Einheiten erstellt werden. Erstellte Einheiten können im aktuellen Zug nicht gezogen werden.

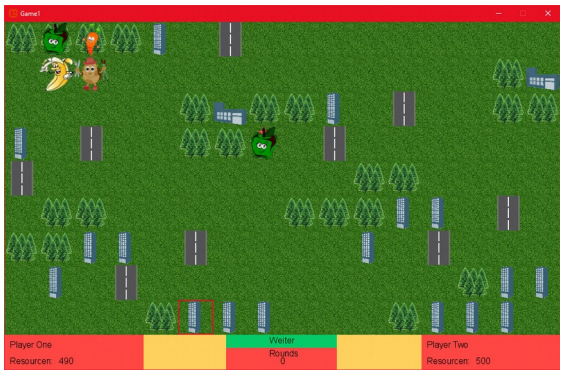

ABB3:Hier wurde der Apfel ausgewählt und der entsprechende Kaufbetrag wurde Spieler 1 abgezogen Diese Spielfigur kann jedoch erst in der nächsten Spielrunde bewegt werden.

ABB4: Anzeige des Bewegungsradius und eines möglichen Kampfes.

Jede Spielfigur hat eine gewisse Reichweite, unter der sie sich auf benachbarte Kacheln bewegen kann. Sie wird angezeigt, sobald auf eine Einheit des aktuellen Spielers geklickt wurde und diese sich in diesem Zug noch bewegen darf. Durch das Klicken auf eine gelb markierte Kachel kann die Figur auf diese verschoben werden. Rot markierte Kacheln zeigen einen möglichen Kampf mit einer gegnerischen Einheit. Kämpfe sind immer vor und nach dem Ziehen mit direkt anliegenden Feinden möglich. Begonnen werden Kämpfe durch anklicken einer rot markierten Einheit. Im Kampf werden beiden Einheiten die Angriffspunkte(inlusive kleiner Zufallsabweichung) der jeweils gegnerischen Einheit abgezogen. Die

Berechnung erfolgt nacheinander, besiegt also der Angreifer den Verteidiger, indem er dessen Lebenspunkte auf 0 reduziert, dann erhält der Angreifer auch keinen Schaden.

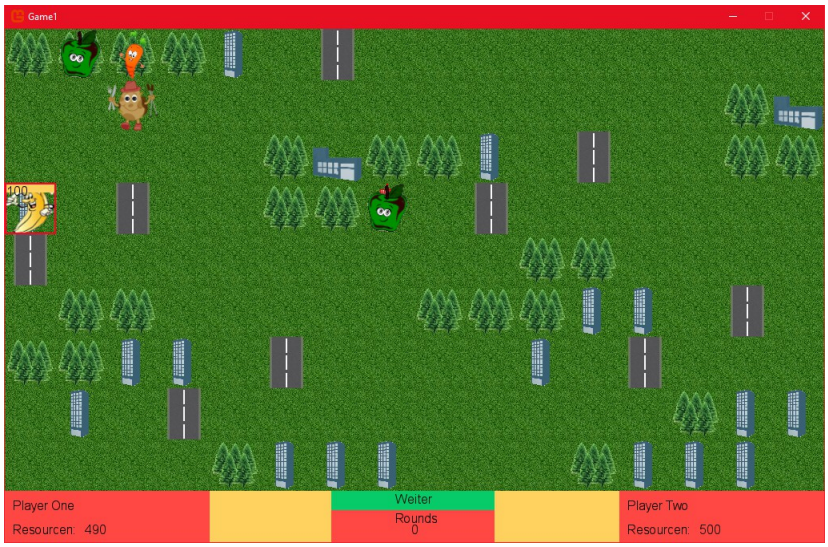

ABB5: Anzeige der Lebenspunkte und Angriffskraft einer Einheit.

Sitzt der Cursor auf einer Einheit, dann werden die Lebenspunkte und die ungefähre Angriffkraft angezeigt.

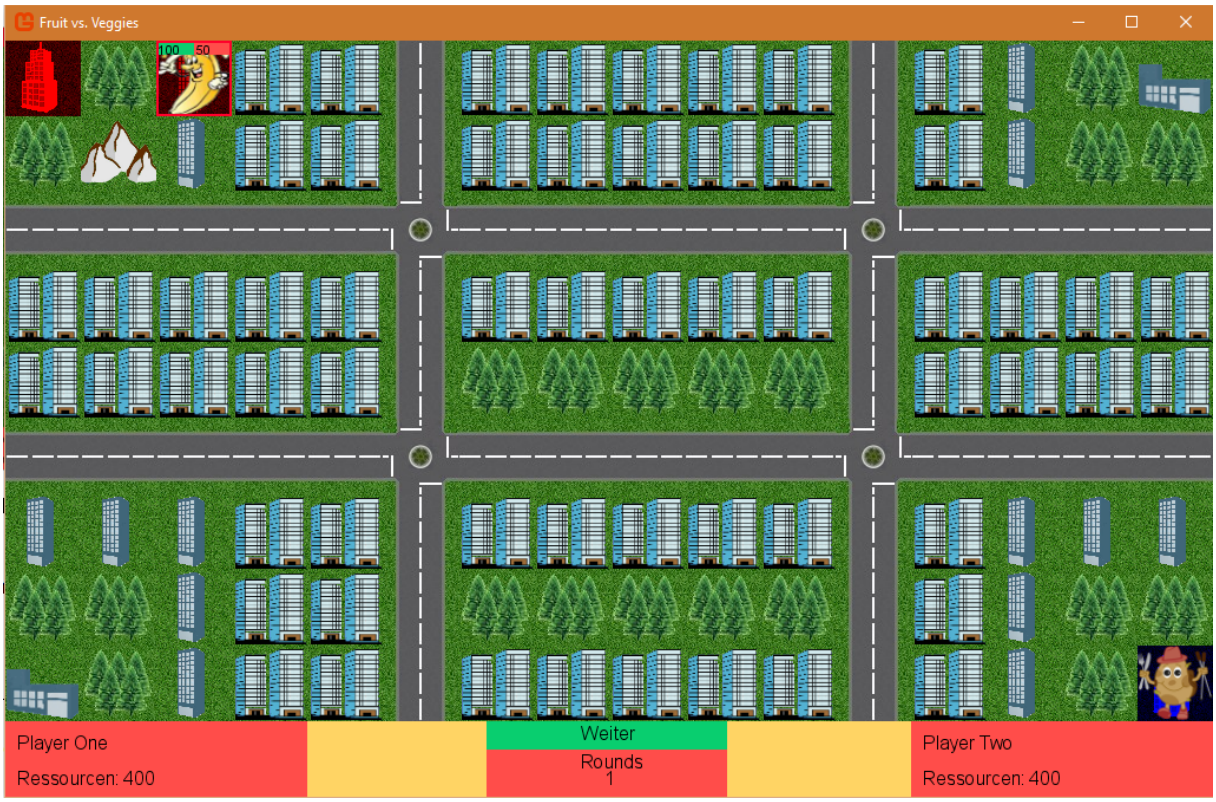

ABB6: Betritt eine Einheit eine übernehmbare Kachel, so wird sie zum Besitz des kontrolierenden Spielers. Die Kachel ändert ihre Farbe entsprechend der Hauptbasis des jeweiligen Spielers.

Am Anfang jeder Runde erhält der aktuelle Spieler von jeder Ressourcen-Kachel in seinem Besitz eine gewisse Menge Ressourcen.

Ein Spiel gilt als gewonnen, sobald ein Spieler alle Hauptbasen des Gegenübers übernimmt.