Tetroulette

Project Plan for
Professional Issues and Ethics
(COMP4920)

By Tim Humphries, Godlin Rajendran, Michael Zhou,
Hamza Shoaib, and Calvin Tam.

Introduction

Team Members:

Tim Humphries, Godlin Rajendran, Michael Zhou, Hamza Shoaib, and Calvin Tam.

Project Brief:

Tetroulette - an online multiplayer, co-op Tetris game.

Inspirations:

The following are existing games that provide inspiration for this project. The main similarity is the sense of a real-time view of the other Tetris games that other players are in, while the main difference is that the game objective is not to defeat other players, but rather becomes a cooperative survival game. The concepts and differences will be examined in more detailed in the following section.

Screenshot of two different Tetris variants that demonstrate multiplayer functionality

Background

Tetroulette is a browser-based multiplayer co-operative puzzle game, inspired by the popular tetra-block puzzle, Tetris, and novelty communication tool Chatroulette, which randomly matches users together for conversations. A significant number of Tetris games take place at once. However, unlike most multiplayer Tetris adaptations, players are only able to make one consecutive move; after this, they move on to a different game board.

Tetroulette is a variant of the Tetris formula which distinguishes itself by that fact that it does not focus on individual point scoring, rather the aim is to keep the games alive as long as possible. At its best, the game aims to create a sense of constant communal crisis. The pleasures of the game lie in working together successfully with a large number of strangers, as well as the flow-state fun of a game that does not halt. Additionally, live Tetris streaming on a large scale has not been done in a browser before, to the best of the team's knowledge. Playing tactics will need to differ significantly from typical advanced Tetris strategies, as other users must understand the plan on sight.

The scope for the project begins with a working single-player Tetris frontend, and continues to include live streaming of games to other clients (spectator mode), full collaboration and game rotation, and (ambitiously) community interaction through a small web portal.

Methodology

Agile Approach

The team will be employing an Agile approach for this project. Agile was primarily selected due to the difficulty in defining the entire scope and completion time of the project, which would require successive iterations to reevaluate certain features and the general direction of development. Frequent user feedback will also be instrumental in the development of a high-quality product. The particular Agile methodology that will be used is Scrum, which focuses on iterating and delivering incremental releases.

Using the Scrum Methodology

Due to the short period of time available (~6 weeks), the team has decided to do weekly sprints to achieve the maximum number of iterations. The aim is for an iterative working release after 1-2 sprints. Now to achieve this, the team has set out a rough plan which consists of:

- Scrum meetings - Main meeting time is Sunday. Not feasible to do daily scrum meetings, will do asynchronous communication via Facebook due to differences in availability within the team
- Sprint Planning - Week 9 and subsequent changes as the project progresses
- Sprint Review - Every Sunday
- Sprint Retrospective - Every Sunday
- Scrum roles designated
    - Scrum Master - Calvin
    - Product Owner - Tim
    - Developers - All team members
    - Testers - All team members

To guide the team on tracking current progress and an agreed-upon backlog, the team will be using the Pivotal Tracker Project Management Software.

Testing

In this project, testing provides a feedback loop on both the code quality and the user experience. On the front-end side, testing will focus on ‘customer’ feedback, essentially performing functional testing with the product owner interacting with the website to ensure the acceptance criteria of the user stories are met. Play-testing on each iteration with a sample group of Tetris fans will also provide some feedback that will allow the developers to improve on. On the other hand, on the back-end side, the team will be using unit testing as a means of increasing confidence on the game logic and web backend. The developer does not test his own code and this is to ensure a broader set of consensus is achieved before it is regarding as part of the core master branch of code that everyone will rely upon.

Continuous Integration

Another commonly employed technique used in Agile is Continuous Integration (CI). To achieve this, the team will be using git for version control. The approach will be to merge early and often, which will run against automated unit testing for all commits to master. The master branch will also be tested at a functional level with every major change to ensure the master branch is stable. For dependency management, JavaScript libraries stored within the source tree and version numbers of Python modules documented.

Design

System Architecture

The system architecture consists of 3 distinguished parts: 1) a pure game state library (doesn't interact with networking, gui, etc.) that will contain an identical implementation in JavaScript and Python, 2) the front-end that deals with the presentation of the game state of Tetris and 3) the back-end which will act as the controller as per the MVC design pattern.

1) Game State Library

Presents an interface that allows reading of the entire state for display or remote sync as well as pushing in new blocks or user input, like translation and rotation requests. It works on a tick-based system, where every input for a frame is pushed in, followed by a call to update(), which will apply all changes and cause further input to be associated with the next frame.

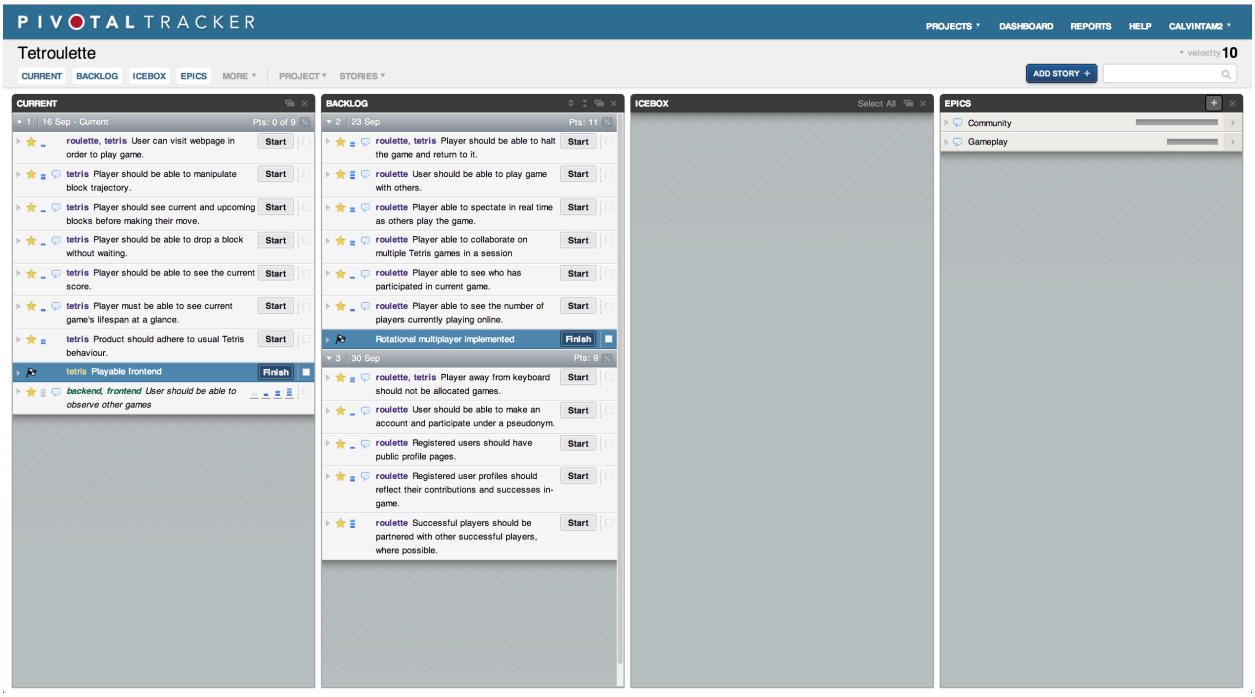
2) Front-end

As well as the usual decoration, on every frame, JavaScript will pull data from the game state and display it by colouring the cells of a HTML table (subject to change). It will also receive input events of other games from the backend through SocketIO and update the respective local game states.

3) Back-end

The back-end will take care of the business logic of the website, the game engine and all other interactions. It will be written in Python and will perform real-time communication with the front-end using SocketIO. It is responsible for tracking game states so new players can join, as well as forwarding input events, so every player is seeing the the same games.

User Stories and Product Backlog



Screenshot of User Stories, Product Backlog and Release Plan organised in Pivotal Tracker

The project is split into three epics and about 20 user stories, to be pursued in the prioritised order presented below. Stretch goals are displayed in **bold**; these are our ambitious targets. The Acceptance Criteria documented are minimums for each story.

Each item in the backlog has been allocated story points; these are estimates of difficulty. An item with one story point should take up no more than two hours of a single developer's time, implying another hour of testing and time for code review. The initial estimate is around 20 work hours for the items in the first epic below, with those hours distributed across the team. There are also less functional matters, like front-end design, which will consume a considerable amount of time and at the same time difficult to estimate.

Epic	User story	Acceptance Criteria	Story Points
I want to play a game.	As a player, I want to visit a webpage to play the game.	Visiting some URL should start the game.	1
	As a player, I want to manipulate block trajectory.	When I press the left and right arrow keys, the block will move left and right respectively.	2
	As a player, I want the ability to rotate and drop blocks.	When I press the up arrow key, the block will rotate clockwise. When I hold the down arrow key, the block will drop instantly.	2
	As a player, I want to see the current and upcoming blocks before I make my move.	While a game is underway, a clear display shows the current falling block, and the next block to fall.	1
	As a player, I want to see the current game's lifespan at a glance.	While a game is underway, a clear display shows the game lifespan in the correct units (minutes, seconds) that updates in real time.	1
	As a player, I want to see the current score at a glance.	While a game is underway, the current score is clearly and prominently displayed. It updates in real time as I play.	2
	As a player, I want to be able to pause gameplay and return to it later.	There is a clear interactive visual element that allows me to stop playing momentarily.	2
I want to play with others, online.	As a player, I want to be able to spectate in real time as others play the game.	Visiting the site, I can click a button to view other games played in real time without actually playing.	3
	As a player, I want to collaborate with others on multiple Tetris games in a session.	After I place a tile, I am immediately given a different game to play. Other players operate simultaneously on the same set of games.	3
	As a player, I want to see who has participated in my current	While playing, I should see a list of players who have recently	2

	game.	made a move on the current game.	
	As a player, I want to see the number of users currently playing online.	When visiting any page in Tetroulette, a non-obtrusive counter should be visible, displaying an estimate of the number of current players online.	1
	As a player, I want other players away from the keyboard to be excluded from live play.	**When I do not interact with the game for some period of time, the 'pause' functionality should be triggered, so I do not affect multiplayer games.**	**3**
I want to participate in a gaming community.	**As a user, I want to be able to make an account and participate under a continuous pseudonym.**	**Before playing, I am able to sign up and enter a pseudonym, used for player listings and profiles.**	**2**
	As a registered user, I want a public profile page that reflects my contributions and successes.	**When not playing, a section of the website allows me to browse other user profiles, which contain various success metrics.**	**2**
	As a successful player, I want to avoid antisocial or unskilled players.	**When a game's lifespan passes a certain threshold, players with poor records should be excluded from moving on it, unless there is nobody else available.**	**3**

The initial sprint will target the first epic in its entirety. It is estimated one or two seven-day sprints should be sufficient for the first epic, at which point a release is planned for in order to receive feedback.

The maximum timeframe for the second epic is four sprints; however, if things work smoothly then it could be completed within a single sprint. In the usual Agile fashion the team will use the best of its current knowledge to adapt to the situation as it presents itself. The plan is to have the second release available after the non-bold user stories have been minimally satisfied.

If time remains after the second release, the stretch goals in the third epic will be mixed with bug-fixing, chores and other improvements. We may have intermediate releases when challenging functional goals are fulfilled.