Multiplayer Social Gaming NYU Course (http://goo.gl/G8QNcQ)

Office hours

Office hours are before the lesson, email me before so I can plan accordingly.

CSCI-GA.3033-007
(1145)
Special Topic: Social Multiplayer Games
Yoav Zibin
R 5:10-7:00PM
60 Fifth Ave C04

Grader/Helper:
Amanpreet Singh
amanpreet@nyu.edu
apsdehal@gmail.com

Short bio about the lecturer

Yoav Zibin (yoav.zibin@gmail.com) did his PhD in the Technion, a postdoc in MIT, worked in IBM research, and now in Google NYC working on video ads. He co-founded and worked for 3 years in a multiplayer gaming startup called Come2Play.

Short course description

Note: the content will be different from previous semesters.

I will teach about web technologies such as HTML5, JavaScript, TypeScript, AngularJS, Firebase, Grunt, and testing using Karma, Jasmine, Protractor.
The course will also teach about artificial-intelligence for games, ranking systems,
and integrating with social platforms.

We will build an open source multiplayer gaming platform and many games.
The platform has two components: **GameBuilder** and **GamePortal**.
GameBuilder is a site that allows us (the students) to create games.
GamePortal is a place (sites&apps) where users can play all games with their friends.
GameBuilder will have just two versions (one with ReactJS and one with Angular2).
GamePortal will have many versions, e.g., some teams will create a web version, while others android/ios versions, and some may choose other technologies such as React Native, see our spreadsheet for list of possible versions.
Both GameBuilder and all versions of GamePortal will connect to Firebase, which will contain our database (of images, game specs, user data, etc).
See the team HWs for GameBuilder and GamePortal here.

Game spec and firebase DB will evolve collaboratively during the course.
Each week we'll add new games to our database, and each team will slowly add more features to their version of GameBuilder/GamePortal.

Version 0 of GameBuilder that the grader built

https://yoav-zibin.github.io/GameBuilder/

Vision for GamePortal

GamePortal is a place (site&app) where you can play almost any game with your friends in a realtime fashion, while video chatting with them.
Example use-cases:

1. **One-on-one (two players)**: I open the app, click play with my father, he gets notified and opens the app, and we video chat while playing chess/backgammon/etc.
2. **Group game**: It's time for our **poker** night! I open the app, click on our existing game, send a message because some people are still not in the game (they get a notification and either go into

the game or reply that they can't make it tonight), we video chat while waiting for some of the players to show up, and then we start the game.

All real-life games (i.e., games you play in the real life, without a computer) have a couple of **common elements**: board, pieces, cubes, cards (or pieces with two faces like in scrabble), sheets of paper (that you can write/draw on), etc.

- GamePortal won't implement the rules of the games (the players will need to know the rules and enforce the rules themselves).
- GamePortal will just have the graphics and logic for the common elements. Any player can move the elements around and perform common **operations** on them.
- There is always a **board** (some background image of a certain width-to-height ratio).
  Double clicking (or pinching) on the board will zoom in and out.
  Long press to select a rectangle on the board, which will select all the elements of the same type within that rectangle (and then you can do some operation on the elements, like shuffling them, moving them around etc)
- Some elements can support **dragging**, i.e., the players can move the element on the board.
- Clicking on a **element** does an operation that depends on the type, e.g., rolls a cube, flipping a card, etc. An element may have a menu if it supports multiple operatins, e.g., in a game of poker, the menu for a card may show options to shuffle all cards, privately flip the card (maybe also show the card just to a specific player), or show the card to all players.
- The **players** will be in a video chat with one another and will see all the operations in real-time.
  The players will also see all the clicks, drags, etc.
  Show touch screen events (each player has a color, so you see who is doing the drag).
- Each player will have a rectangle showing it's **video stream**, with an option to mute video/audio. If video is killed, then we just show the avatar (with an indication to whether the player is online/disconnected).
- Each player will have a **private area** (cards that are in the private area are shown only to that player, while the opponents just see the picture side). In other games (like Stratego) each piece may be public or only visible to certain players.
- Optional advanced features: sky is the limit.
  - **Non-real-time** mode: the players can still perform operations (move elements etc) even if some players disconnect. When those players connect the next time, GamePortal will **replay** the stuff they missed. This way you can even play the games in a **ping-pong fashion**, i.e., you make your move, send some chat message that you're done (or maybe we'll add an element that switches turns), and the opponent can make their move.
  - Another idea is a **community game**, where anyone can move the pieces or start a new game, there is a public chat (with optional video/audio). It's a way for you to see who's online now, what moves different players did, etc.
  - **Timeline**: the entire game is recorded (with a history timeline, kinda like in google photos, so you can see the entire history of a game + chat, in one place).

Lesson breakdown

1. Course overview.
   Game spec: the difference between the game and platform.
2. javascript, JSON, TypeScript, working with Git and Github
3. HTML & CSS including HTML5 such as animations, drag-n-drop, flexible-box-model.

4. Firebase
5. JS platforms: Angular, React, etc.
6. Unit testing
7. Artificial Intelligence (AI) for games. Heuristics, A* search, alpha-beta pruning, etc.
8. PhoneGap
9. FB social features.
10. Firebase analytics & Google analytics
11. WebRTC
12. Monetization

Grading

For building the platform, you will work in teams of 1-3 students.
For creating the games, you will work alone.
The final grade will be 50-50, i.e., half the grade will be for the team platform project and half for the individual games.

The deadline of the platform HWs is 24 hours before the next class (so Wednesday 5pm). A HW is submitted by committing into a public GitHub repository.
The 24 hours before the course will be used for finding bugs.
Each HW will be graded, and your final grade will be the average of all HWs (except the one with the lowest grade).

Finding bugs

You can find bugs in anyone's work and get some bonus points (I'll decide the amount of points in class depending on the severity and complexity of the bug).
You have to create a github issue for the bug.
Make sure that a similar issue doesn't exist already.
Include a youtube **video** link where you show how to reproduce the bug twice from initial conditions (to ensure the bug is reproducible). I like to use this chrome extension for creating such videos and uploading them to youtube.

HW1

Join our Google group nyu-gaming-course-fall-2017.
If you have any problems or questions, send an email to our group:
nyu-gaming-course-fall-2017@googlegroups.com

Create your first square board game!
(Eventually we could play these games in our GamePortals.)

Choose a board game where there is no hidden information (so don't choose a game like scrabble).
For example:

- Y-game

- Xiangqi
- Shogi (you can use these board graphics)
- Pacru
- Petteia & Latrunculi
- Chaturanga (Indian chess)
- Shatranji (Persian chess)
- Janus chess
- Zèrtz or Dvonn or Relax
- Pasang
- Connect 4
- Connect5 (Gomoku)
- Connect6
- Kamisado
- Go/Weiqi
- Reversi
- Lines of Action
- Bōku
- Chinese checker
- Game of the Amazons
- Janggi
- Nine Men's Morris
- Chess
- Halatafl
- Hex
- Havannah
- Biloba
- Mancala (Oware)

As the course progresses, our GameBuilder versions will get more features and it'll be able to support more game types such as games with dice:

- Barricade
- BackGammon

And eventually also card games and complex games with hidden info:

- 90,000 board games in https://boardgamegeek.com
  See:
  https://www.nature.com/naturejobs/science/articles/10.1038/nj7663-369a
  https://boardgamegeek.com/filepage/142581/pdf-xtronaut-rule-book-2nd-printing
- Banqi
- Stratego
- Scrabble

- Rummikub
- Risk
- Luzhanqi
- 3-players games, e.g., 3-players Chess
- Game stutdents did in previous semester (old, new)
- Diplomacy, monopoly
- DiceWars
- http://en.boardgamearena.com/#!gamelist
- http://www.vassalengine.org/wiki/Category:Modules
- TripleA
- OpenFracas / Risk / **Age of Conquest**
- Pioneers / The Settlers of Catan
- Puerto rico
- A complicated board games
- More board games: http://www.mindsports.nl/index.php/arena
- Card games, e.g., Poker, Taki, Rummy, Continuo, bridge, gin, rummy, spades, hearts, uno, crazy eights, Euchre, phase 10, Cribbage, nine card, canasta. Mahjong Durak, thousand, nine, Klondike, spider, freecell. Batak, mau mau, buraco
- Anything in this picture - board games heaven at Google :)
- Mafia (The Resistance or One Night Werewolf)
- TwixT
- Puzzles
- A game *like* DrawSomething, Words with Friends, Matching with Friends, Hanging with Friends, Farkle, Trivial persuit. E.g., like DrawSomething, but instead of drawing just words, you draw movie titles (you can get a list of the most popular 1000 movies).

Lesson2: JavaScript and TypeScript

Show the games (use random permutation to select the order).

Talk about the different framework for the team HWs (see our spreadsheet):

- WEB: no framework
- WEB: ReactJS (with optional redux)
- WEB: Preact
- WEB: Angular 2 with TypeScript/Dart
- WEB: Polymer
- Android: in Java/Kotlin
- iOS: in Swift
- Native cross-platform: unity
- Native cross-platform: react native
- Native cross-platform: Xamarin
- Native cross-platform: Appcelerator
- Native cross-platform: flutter

If you want a different framework, add to to our spreadsheet.
The most important thing is that you can connect to firebase from that framework.

Intro to JavaScript (show in console).
Always use let (let vs var)
Optional: Expand your JavaScript knowledge, e.g., watch JavaScript Lecture by Douglas Crockford.
Basic TypeScript.
TypeScript slides.
Show Visual Studio Code, install typescript:

```
sudo npm install -g typescript
tsc --watch
```

Show GamePortal project and how to surf to it on github.io

- Working with Github, Github pages, Github features

Working with GIT:

- Basic Git commands
- Git tutorial
- Simple Git guide

HW2

Create game #2.

Choose the version of GamePortal/GameBuilder you want to work on: enter your name next to exactly 3 possible versions in our spreadsheet. I'll create teams accordingly.

Install npm and NodeJS, see instructions.

Editors for web development

You can use any environment/text editor you want, but I highly recommend Visual Studio Code.
Other options are: Atom with TypeScript package, webstorm (student license), eclipse, Sublime2, NetBeans.
We will also work with github, so I recommend github for mac or github for windows.

Lesson3: CSS & HTML

- Intro to CSS&HTML: slides.
- jsfiddle, box model, dimensions, block vs inline, positioning (tutorial).
- SVG, intro, example, SVG backgammon graphics, chess, Risk, XiangQi.
- HEX boards
    - Hexagon board example.
    - Explanations: Hex grids, Hex math

    - Example using canvas: canvas example.
    - Example using svg

Lesson: Firebase

- Firebase for the web
    - Reading&writing data
    - Presence example
    - Transactions in Firebase , example

AngularDart
Get started, install guide for MAC
WebStorm has a free student license, but you can also use VisualStudio extension for Dart

Lesson: mobile UI

- Make sure to setup your Android device for debugging.
- Debugging mobile graphics: on android, on ios.

Testing your mobile site:

1. Emulate a mobile device on chrome using the emulation panel.
2. Use it on a real smartphone

Example WEB UI frameworks:

- Polymer
- Angular material: Getting started, layout (flexible-box-model), Material design icons.
- bootstrap CSS, BootStrap components, Angular BootStrap
- Ionic framework , Ionic with Sass
- Angular UI

Lesson: animations, drag-n-drop, offline playing, tools.

Useful HTML5 Resources: HTML5 rocks, Dive into HTML5, Google Web Fundamentals

- Appcache: HTML5 manifest, Gotchas.
    - Service workers (like appcache, but better; appcache on http is deprecated, on https still available)
- Animations: CSS transform, transitions and animations. animations-example.html.
    - ng-repeat animations, testing animations (playing with cubic-bezier).
- HTML5 animations API,
    - It's not available on iOS or Android 2-4
    - But there are polyfills (so it works, with that polyfill, on iOS and old Android as well)
- Drag-n-drop without jQuery: click-to-drag, drag-pieces (code), area map drag-n-drop (area map tag help), example for irregular boards in game chinese-checkers (code uses `document.elementFromPoint`)

- Minified and Obfuscated code
- Tools:
    - npm , Jasmine, Karma, Karma-coverage + Istanbul
    - UglifyJS
    - Grunt - JS task runner
        - UglifyJS + concat - minifies your JS files
        - PostCss (autoprefixer + cssnano) - minifies your CSS and add vendor prefixes
        - ProcessHtml - processes your HTML files (so they'll load the minified JS files)
        - manifest - creates an appcache manifest with a timestamp.
        - karma - runs unit tests
        - watch - Automatically run unit tests on file-save:
          `./node_modules/grunt-cli/bin/grunt karma:unit:sta watch`

Lesson: Artificial Intelligence (AI)

Artificial Intelligence (AI) for games. Heuristics, A* search, alpha-beta pruning, etc.

- Presentation: Min-max search, alpha-beta pruning, heuristics (evaluation of states and ordering of moves).
- Optimal decision tree for player X or XKCD version
- XKCD AI and Solved games
- alphaBetaService.ts
- Neural networks: beating the best Go player and 4-1.

Lesson: unit testing and e2e testing (selenium/WebDriver)

Testing (slides).

Read more on software testing.

Demo tools and frameworks used: Jasmine, Karma, Karma-coverage + Istanbul.

- End-to-end tests (using protractor).
- Finding elements: by.css, CSS selectors
- Protractor (Protractor docs)- E2E test framework for AngularJS
    - WebDriverJS - E2E test framework for JS
    - http-server - Simple HTTP server
    - Grunt:
        - CanIUse.com
        - PostCss (autoprefixer + cssnano) - minifies your CSS and add vendor prefixes
        - Http-server
        - Protractor-runner
        - Optional: Protractor-coverage

- protractor&appium
- continuous test integration

You can also run protractor directly (not via grunt):

1. `./node_modules/http-server/bin/http-server --ssl -c-1 --cors -a localhost -p 9000`
   `Or`
   `http-server -a 127.0.0.1 -p 9000`
   Test the webserver work by going to: http://localhost:9000/index.html
   (don't kill the server!)
2. `./node_modules/protractor/bin/webdriver-manager update`
   `./node_modules/protractor/bin/protractor protractor.conf.js`

Lesson: Material design

Material design

- Material design intro
  - Color, language, spacing, FAB, tooltips
- Material UI in ReactJS
- Angular material: Getting started, layout (flexible-box-model)
  - CodePen: grid, FAB
  - Watch this video about angular material start project (there many other videos if you like).
- Material start project
  - My simplified clone of material start (on gh-pages)
- Material design icons (other material icons)
- Resizer
- Responsive UI: responsive UI in material design

Lesson: Misc

I18N all your games that have text (also check whether your images need to be localized).

Add screens showing instructions and rules for your games (with relevant printscreens or animated gifs).

Make sure the I10n of the platform is correct and makes sense in languages you know (english/chinese/spanish/french)

Elo ranking system.

Useful FB tools:

- Test console
- Graph explorer (try "/me/friends"), Graph API reference
- OAuth2 intro, FB access tokens, XAuth

Google Play Multiplayer Game Services

Talk about identity+social using Social networks (FB, G+, Twitter) or phone contacts (Twitter Digits and Mobile graph).

Ads: AdMob (making money from ads and promoting your app)

Advertising, marketing (social marketing, e.g., creating FB page and boosting posts, or creating twitter presence and stream search).

Security: OAuth2, web security, XSRF, CSP, mXSS.

Google Web Designer.

Other stuff:

- HTTP , Running a local HTTP server:
  ```
  sudo npm install http-server -g
  http-server -a 127.0.0.1 -p 8887
  (This is not that secure: python -m SimpleHTTPServer 8887 )
  ```
- telnet

Lesson: WebRTC

- **WebRTC** (for video&audio chat):
  - Android: supported natively on Android (for older devices we can release a heavy version with crosswalk)
  - iOS: there is phonegap plugin for iOS
  - Server-side: 90% of users won't need anything except a (publicly available) turn server. For the other 10% we can initially use a service like **xirsys.com** , or install collidor on compute engine.

- WebRTC on HTML5rocks
  - STUN server:
    iceServers: [{ url: 'stun:stun.l.google.com:19302' }]
  - (If you want, TURN servers can be rented in http://xirsys.com/ , or hosted on Google Compute Engine, see CubeSlam code)
- WebRTC signalling using Firebase example
- WebRTC data channels
- WebRTC, chrome://webrtc-internals, PeerJS, Demo
- Another WebRTC Demo on AppEngine
- Record WebRTC & YouTube
  - https://github.com/muaz-khan/RecordRTC

- https://developers.google.com/web/updates/2016/01/mediarecorder?hl=en
- Have an option to record the entire video chat (the two streams, including the game overlays) and upload it to YouTube (your own YouTube channel about your game).