Tower Defence Game

ToDe

(Toadie)

by iomods.com

Tower defence is a closed-source community driven platform for providing a tool capable of driving massively online realms, as well as a rock-solid foundation for a diverse range of projects.

Priorities: LOW, MED, HIGH

Storyline

In a land far far away the queen of cena sniffed the air with anticipation as she knew breakfast was being served when suddenly she exclaimed “oh my! wait wot... that’s not bacon! that’s lamb and an incoming army of the undead transgendered aliens with body armour and powerful weapons”. DO YOU SMELL WHAT THE ROCK IS COOKING?!

Overall Game Premise

- Leveling up can mean teleporting to a more elite game server while retaining some of your ill gotten gains from lower levels.
- Expensive but ability to purchase base size upgrades
    - Also able to purchase more plots of land nearby
- Capturing places
    - A conquered base location can be occupied to increase the economy of the player, until player is complete killed or frees himself
    - Need minions to kill checkpoints and enemy bases
        - checkpoints for points, give more resources to push against other players
        - Checkpoints will have defense as well, low constant towers meaning a big push of minions are needed to damage it
- Balance between attack and defence, sending minions takes resources needed to create turrets
- Exponential difficulty to prevent people from rolling over people
    - Army maintenance costs!
    - Decaying bank account
    -
- Fog of war (see game server specifics)
- Teams? This would be interesting.
    - Players prefer FFA in a lot of situations, and people love being a “lone wolf”
    - Separate game modes, FFA, Teams, competitions, etc.
    - Teams **x4**

User experience

- Clean design?
    - This means no grass background, that shit ugly ← i will kill you ← No, you won't.
- Expando-menu to drop items, buildings, minions on a cooldown timer
    - Expando retracts to show full area unobstructed
    - Automation
        - Instead of clicking on e v e r y s i n g l e individual item for upgrading, have an /upgrade all/ option and/or equivalent.
    - Menus pop out from a sensible location
        - Ex; clicking on the base may open a menu at the base location
        - Ex; clicking on a turret for upgrading will open upgrade menu AT turrent location not someone where the other side of the screen

Mobile devices

- Transparent joystick-like controls for navigation.
    - Scroll like a map, open expando-menu to drop items/minions.

Game server specifics

- Send data only for items within view-pane
- View-pane grows when you send out more minions
- Minions belonging to groups of people to ignore each other
    - Or attack synchronized but immune to each other
    - If teams, would only target opposing team/neutral towers, if FFA target all but own

Server Protection

- Tor exit node filtered at firewall
- Block multiboxing
    - 3 games / IP?
- Captcha
    - Shows when something suspicious occurs like multiboxing within the limits?
    - + tokens
    - https://www.google.com/recaptcha/api2/demo?invisible=true

Technologies

Streaming servers

- Connect to youtube, twitch.
- Connect to game server.
- Stream a delayed game map.
- Much more efficient to somehow generate image from data and stream that, versus rendering in a client and then screen-grabbing that

Game servers

- Debian based linux distribution
- Node.js
- Secret Auditorium mode for our streaming servers.
- Our own provisioning, monitoring and restarting platform.
- Archived game-state across multiple servers
    - Say a server dies and players disconnect
        - Once player is reconnected, the game-state is restored from the archive despite being a totally different server.
        - This means a crashed server on the network is technically restored without losing player/realm state
        - Crash may not be possible to do that, but transfer to another server with data would be good basis for the levelling/world-hop implementation
        -

Web servers

- Debian
- Nginx
- Static files
- Behind cloudflare

Website design

- Social media links
- Push youtube subscription in people's faces - my biggest regret with agariomods was never building a youtube channel.
- **<insert login system here>**
    - You wanted patreon, right?
    - Maybe later. Im gonna try to be more reasonable.
    - Unless someone says its ok? :P
    -
- In-game chat
- Friend list / instant messenger with irc gateway
- Skin service for customising play's base and minion appearance
- Global outside-game chat
    IRC and some kind of web gateway

Framework documentation begins on the following page.

Tower Defence Framework

Internal documentation

as of April 4th 2017

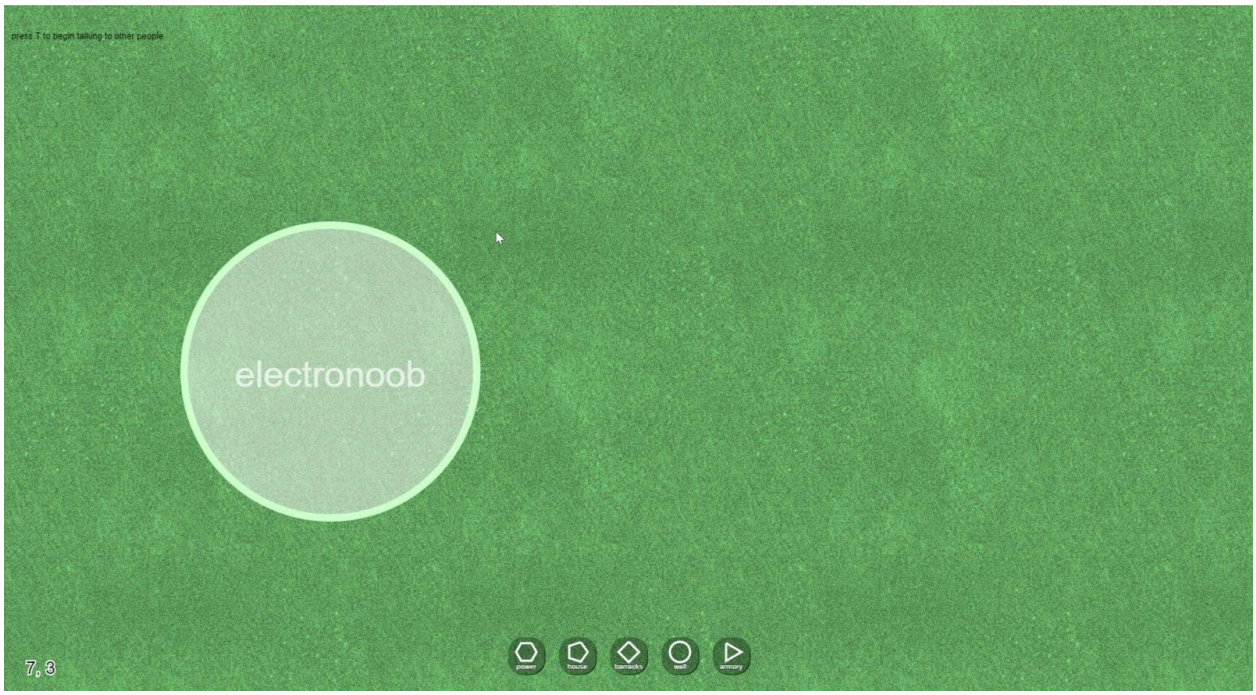

Figure 1. Current progress of the game ui.

Filesystem structure

index.html

Loads the following JS files in order.

vector.js
helpers.js
icons.js
game.js
chat.js

chat.js

Provides a global chat room for players to communicate.
Provides an IRC-like protocol between client and server.

Eslint properties, Exported: **c**. Global: **h**.

Exposes chat class, to instantiate:

Add to index.html:

```
<input id="textInput" type="text" value="" placeholder="">
```

Add to the relevant js section:

```
var c = new chat();
c.init();
```

Basic settings

this.**isTextInputAlwaysOpen** = 0;

If you want the text box for chat input to disappear between message sending then set this to 0, however on the other hand to keep the text box visible at all times then set to 1. If this is set to 0 then the text box becomes visible upon pressing the letter 't' on the keyboard; there is no way to open and close this on mobile devices. Once a message is written the user is to press the enter key for it to be transmitted to the participants, if the textbox is set to open and close between messages then the input box will disappear too.

Server message handling

Currently there is no websocket connection of game server implementation available and as such the chat protocol does not yet exist.

this.**messages** = Array();

Although mostly useless except for debugging at this stage, the messages array can be populated using the following standard for rendering into the UI at runtime.

```
this.messages.push({ username: "electronoob", type: "part", message: "" });
this.messages.push({ username: "electronoob", type: "join", message: "" });
this.messages.push({ username: "electronoob", type: "privmsg", message: ":p" });
```

icons.js

Provides a set a ui button elements with dynamic symbol generation.
Requires an IRC-like protocol between client and server.

Eslint properties, Exported: **c**. Global: **h**.

This doesn't really expose a properly thought out interface.
Add to index.html:

```
<div id="buttonsContainer">
    <div id="buttons"></div>
</div>
```

Initialisation JS:

```
var btnPanel = document.getElementById("buttons");
var btnIcon = {};
```

Basic settings

var **btnIcon** = {};

```
btnIcon.power      = h.getPolyVectors(btnIconWidth/2,btnIconHeight/2 - 5,6,14,0);
btnIcon.house      = h.getPolyVectors(btnIconWidth/2,btnIconHeight/2 - 5,5,14,0);
btnIcon.barracks   = h.getPolyVectors(btnIconWidth/2,btnIconHeight/2 - 5,4,14,0);
btnIcon.wall       = h.getPolyVectors(btnIconWidth/2,btnIconHeight/2 - 5,20,14,0);
btnIcon.armory     = h.getPolyVectors(btnIconWidth/2,btnIconHeight/2 - 5,3,14,0);
```

Setting button width and height here:

```
var btnIconWidth = 50;
var btnIconHeight = 50;
```

Further reading

To understand **getPolyVectors** read the section on **helpers.js**.

vector.js

Provides vector calculations:

sub, add, mag, magsq, div, mul, setMag, hypot, lerp.

Eslint properties, Exported: **Vector**. Global:.

Exposes Vector class, to instantiate:

var a = new Vector(x,y);

In the Vector documentation, ‘a’ is not passed as an argument because ‘a’ is the target Vector whose methods are being used in order to modify it’s properties in situ.

sub = function(**b**);

Calculates new vector position from: a - b.

add = function(**b**);

Calculates new vector position from: a + b.

mag = function();

Returns: the magnitude (length) of the vector a.

magsq = function();

Returns: the magnitude (length) of the vector a, squared.

div = function(**value**);

Calculates new vector position from: a / value.

mul = function(**value**);

Calculates new vector position from: a * value.

setMag = function(**value**);

Calculates new magnitude of this vector from value passed.

hypot = function(**b**);

Math.hypot(a.x - b.x, a.y - b.y)

Returns: the hypotenuse of the vector a in relation to vector b.

lerp = function(**b**, **step**);

*a+(a-b)*step*

Returns a Vector between two Vectors at a specific increment. The step parameter is the amount to interpolate between the two values, where 0.0 is equal to the first point, 0.1 is very near the first point, 0.5 is halfway in between, etc. The lerp function is convenient for creating motion along a straight path and for drawing dotted lines.

Technical specification

1.0 Network protocol between client and server. ***Draft***

Network byte order is big-endian.

As received by client

mnemonic	meaning	opcode	bytes	description
CON	connect	**01**	**1**	**Connected to game server successfully, sent once player is authenticated etc.**
DISC	disconnect	**02**	**2**	**Disconnected from server. (reason)**
JMP	jump server	**03**	**256**	**Jump to another game. (address, security key)**
DIE	died	**04**	**256**	**You died. (reason)**
SPW	server password	**05**	**1**	**Prompt for game server password.**
NPW	name password	**06**	**1**	**Prompt for password needed to use a name.**
UID	user id	**07**	**3**	**Issue user id for session. (Uint16)**
MAPW	Map width	**08**	**3**	**Map width specified by 2 bytes. (Uint16)**
MAPH	Map height	**09**	**3**	**Map height specified by 2 bytes. (Uint16)**
MAPX	Map view x	**0A**	**3**	**Map view location on x axis specified by 2 bytes. (Uint16)**
MAPY	Map view y	**0B**	**3**	**Map view location on y axis specified by 2 bytes. (Uint16)**
MAPID	Map id	**0C**	**3**	**Map or game server ID number specified by 2 bytes. (Uint16)**
CUSR	Create user	**0D**		**Spawn a user on game server.**
CMIN	Create minion	**0E**		**Spawn a minion onto the map.**
PROTO	Protocol version	**0F**	**3**	**Game protocol version. Client expected to quit if different from it's own version.**
COBJ	Create object	**10**		**Spawn a game object, like a gun or base.**
IRCP	Irc protocol data	**11**	**256**	**IRC protocol, for the in-game chatting features. (irc protocol)**
LDRB	leaderboard	**12**		

LDIMG	Load image	**13**	**256**	**Not sure how this would apply yet.**
LDSND	Load sound	**14**	**256**	**Not sure how this would apply yet.**
SWM	Server welcome message	**15**	**512**	**Welcome message to be displayed on client. (Uint16 message)**
SGEL	Server game event log	**16**	**512**	**Broadcast game events. such as player killed by whom. (Uint16 message)**

As received by client in detail:

CON

0x01 sent to client to signal connection to our game server software. This would be sent after completing any authentication processes.

DISC

0x02 followed by reason for disconnection.

Reason table:

- 0x00; triggered idle timeout.
- 0x01; kicked from server by admin.
- 0x02; banned from server by admin.
- 0x03; kicked by vote
- 0x04;

JMP

Packet structure: [1][251][4]
Packet example: [0x03][“websocket.server.address.io:9090\n”,27321]
0x03 followed by 251 bytes for a null terminated string containing a game server address to jump to as well as an unsigned integer of 4 bytes used as a key to identify to the next game server. Purpose: disconnecting from current session and continuing to another but doing so as an identified player to allow for persistence such as, score, size, army whatever we decide.

DIE

Packet structure: [1][255]
Packet example: [0x04][“you were fragged by electronoob’s cockrocket!\n”]
0x04 is sent to signal being killed with 255 bytes for a null terminated string describing how the player died.

SPW

0x05 sent to prompt the user for a game server password, this may be used for private servers or ones requiring a certain privilege to join it.

NPW

0x06 sent to prompt the user for their password, which is used to authenticate them to their desired in-game username.

UID

Packet structure: [1][2]
Packet example: [0x07][12 34]
0x07 followed by 2 bytes are sent to provide the player's current session user id number.
