A4 - Report

COMP 4451 – Game Programming

Group ID assigned by the course staff	18
Group Alias connect.ust.hk email of any one of the members	hmmleung@connect.ust.hk
Game Name tentative name which can be changed later	Steel of Stalin
Group members Student name (connect.ust.hk email address) E.g.: *Pedro Sander (psander)*	hmmleung
	hkleungai

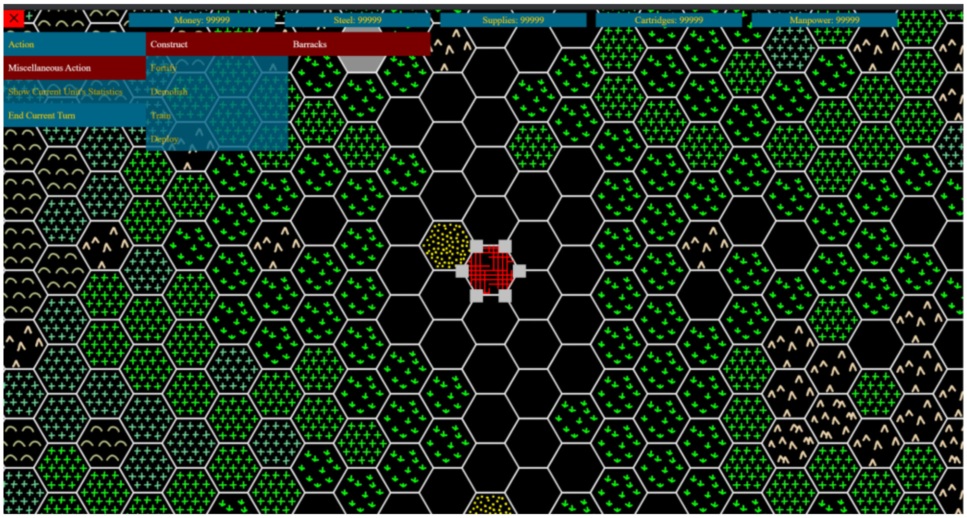
1. Summary of the idea

What the game is about and its overall objective. This can come directly from the proposal if it has not changed, though additions and subtractions are welcome.

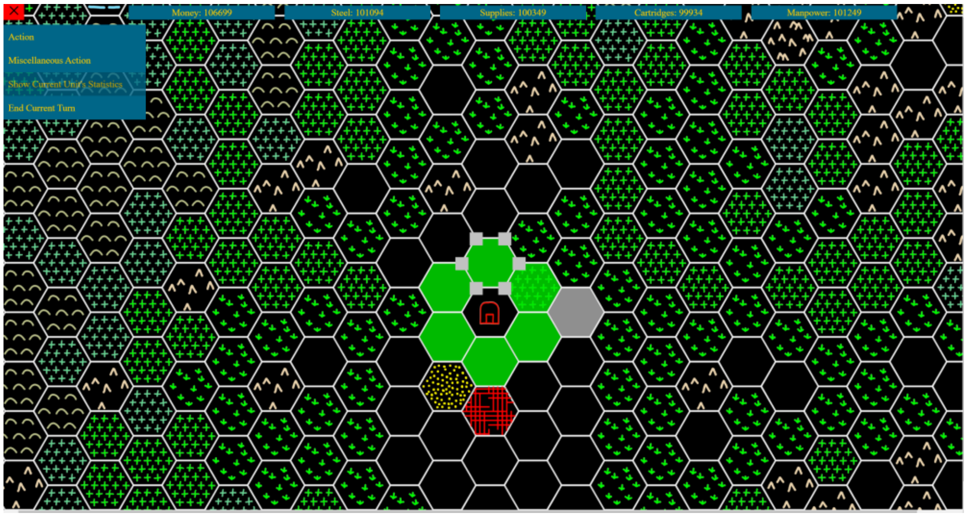
Steel of Stalin is a simultaneous turn-based strategy game. The major objective is to manipulate *units* to capture or destroy enemy's capital. It is about careful planning for *units* to advance through the map, choosing among several advantages. It is also about map control, when players control certain parts of the map or cities to gain advantages.

2. Screenshots

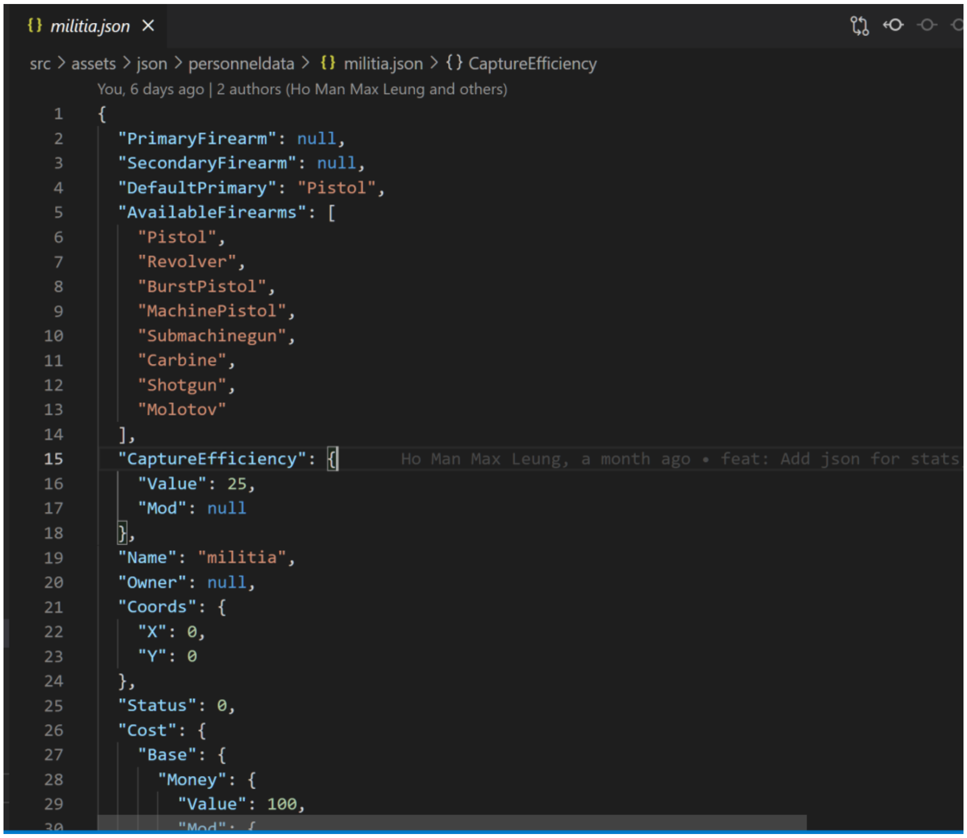
About 5-8 screenshots of your game that best demonstrates your programming challenges. Include a one-sentence description under each screenshot. For example, if you designed a particle system, pick a screenshot that best demonstrates it. There is no need to go into many details. Those should have already been discussed in the video.



Sub-menu highlighting and selection highlighting. Lot of html and scss work for the former, and need to design classes and functions for finding vertices of hexagons for the later.

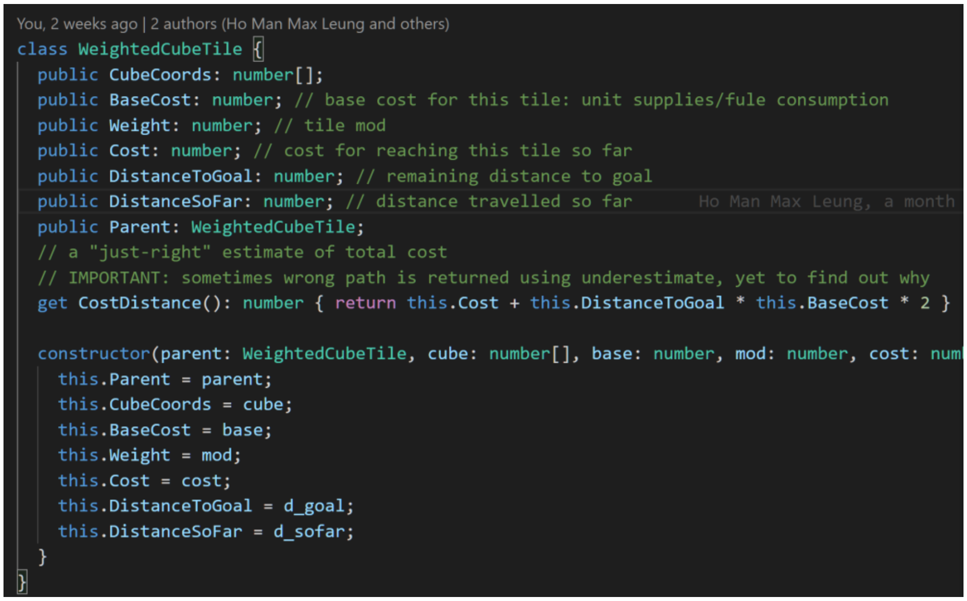


Mouse-hover highlighting. Need to block raycasting for available tiles of some actions (highlighted in green).


{} militia.json ×
src > assets > json > personneldata > {} militia.json > {} CaptureEfficiency

```
You, 6 days ago | 2 authors (Ho Man Max Leung and others)
{
  "PrimaryFirearm": null,
  "SecondaryFirearm": null,
  "DefaultPrimary": "Pistol",
  "AvailableFirearms": [
    "Pistol",
    "Revolver",
    "BurstPistol",
    "MachinePistol",
    "Submachinegun",
    "Carbine",
    "Shotgun",
    "Molotov"
  ],
  "CaptureEfficiency": {      Ho Man Max Leung, a month ago • feat: Add json for stats
    "Value": 25,
    "Mod": null
  },
  "Name": "militia",
  "Owner": null,
  "Coords": {
    "X": 0,
    "Y": 0
  },
  "Status": 0,
  "Cost": {
    "Base": {
      "Money": {
        "Value": 100,
        "Mod": {
```

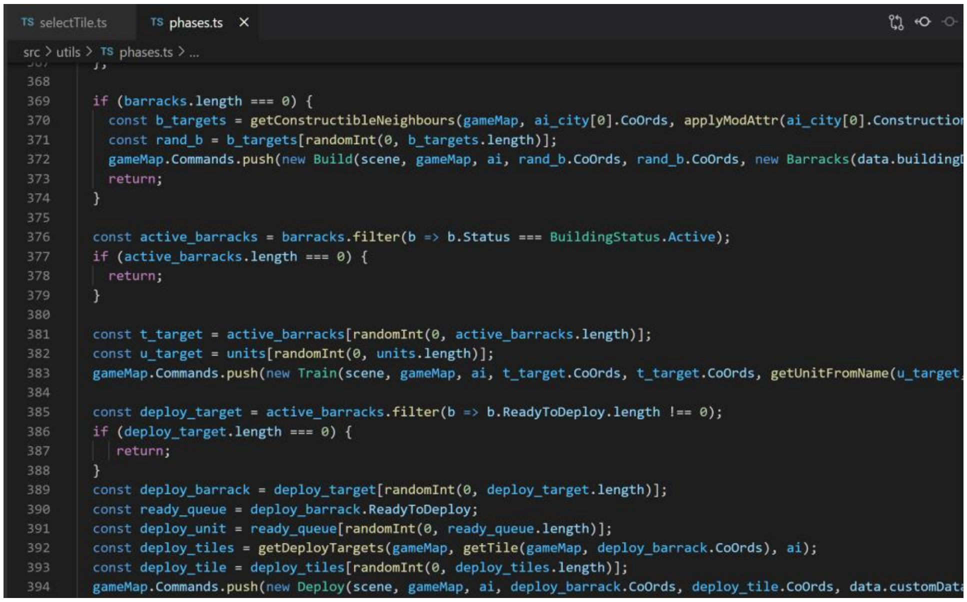
JSON loading and generation. Need to use external codes to generate JSON because TypeScript treats methods of classes as properties and will serialize them as well, potentially causing unwanted effects.



```
You, 2 weeks ago | 2 authors (Ho Man Max Leung and others)
class WeightedCubeTile {
  public CubeCoords: number[];
  public BaseCost: number; // base cost for this tile: unit supplies/fule consumption
  public Weight: number; // tile mod
  public Cost: number; // cost for reaching this tile so far
  public DistanceToGoal: number; // remaining distance to goal
  public DistanceSoFar: number; // distance travelled so far      Ho Man Max Leung, a month
  public Parent: WeightedCubeTile;
  // a "just-right" estimate of total cost
  // IMPORTANT: sometimes wrong path is returned using underestimate, yet to find out why
  get CostDistance(): number { return this.Cost + this.DistanceToGoal * this.BaseCost * 2 }

  constructor(parent: WeightedCubeTile, cube: number[], base: number, mod: number, cost: num
    this.Parent = parent;
    this.CubeCoords = cube;
    this.BaseCost = base;
    this.Weight = mod;
    this.Cost = cost;
    this.DistanceToGoal = d_goal;
    this.DistanceSoFar = d_sofar;
  }
}
```

The node for A* pathfinding. Need to find a suitable bias (2 as shown) to produce correct paths.



```
  };

  if (barracks.length === 0) {
    const b_targets = getConstructibleNeighbours(gameMap, ai_city[0].CoOrds, applyModAttr(ai_city[0].Constructio
    const rand_b = b_targets[randomInt(0, b_targets.length)];
    gameMap.Commands.push(new Build(scene, gameMap, ai, rand_b.CoOrds, rand_b.CoOrds, new Barracks(data.building
    return;
  }

  const active_barracks = barracks.filter(b => b.Status === BuildingStatus.Active);
  if (active_barracks.length === 0) {
    return;
  }

  const t_target = active_barracks[randomInt(0, active_barracks.length)];
  const u_target = units[randomInt(0, units.length)];
  gameMap.Commands.push(new Train(scene, gameMap, ai, t_target.CoOrds, t_target.CoOrds, getUnitFromName(u_target

  const deploy_target = active_barracks.filter(b => b.ReadyToDeploy.length !== 0);
  if (deploy_target.length === 0) {
     return;
  }
  const deploy_barrack = deploy_target[randomInt(0, deploy_target.length)];
  const ready_queue = deploy_barrack.ReadyToDeploy;
  const deploy_unit = ready_queue[randomInt(0, ready_queue.length)];
  const deploy_tiles = getDeployTargets(gameMap, getTile(gameMap, deploy_barrack.CoOrds), ai);
  const deploy_tile = deploy_tiles[randomInt(0, deploy_tiles.length)];
  gameMap.Commands.push(new Deploy(scene, gameMap, ai, deploy_barrack.CoOrds, deploy_tile.CoOrds, data.customData
```

Part of the AI. Need to design the order of considerations for the AI actions to make its decision look more reasonable.

3. Programming challenge specifics

Specify the source files in your submission that include your own code. If you only implemented part of a file, please be specific on the range of lines of code that you implemented. For each range that you specify, describe briefly in one or two sentence what is the purpose of the code in your game. This is important for us to judge the level of programming effort and complexity that went into your game.

Source files that include our own code: All files under the folder src.

- src/assets/*: provides original textures created in blender in png or glb form, all types of json data generated by external C# code, and style layout (scss) of the UI; Many original, creative design here.
- src/flows/*: contains most functional logic that would appear in main.
- src/main.ts: The entry point of the application
- src/props/*: contains class structure of all props (units, buildings, tiles etc.)
- src/utils/*: contains phase logic, loaders and other helper functions
- src/index.html: provides the html template containing classnames for callback and UI

Also there are various linting and config outside src that makes continuous development possible

4. Changes from proposal

Go back to all sections of your proposal and see what you have changed. Briefly describe all the changes (ie., features that you added, or features that you no longer pursued due to time or interest in exploring other ideas).

Features no longer pursued due to time limit:

- Map Creator
- Technologies / Research mechanism
- Logistics mechanism
- Dat.gui no longer in used due to its way of binding data.

Features added:

- Pure js planning -> Nodejs (webpack + typescript) development.
- Interactive UI layout using SCSS technology.

5. Work distribution (group projects only)

Specify the amount of effort of each group member as a percentage (e.g., John 60%, Bob 40%). If you do not agree on the percentage breakdown, please specify separately below what each member believes to be the breakdown. You may further elaborate briefly on what each member has done.

hmmleung: 60%

hkleungai: 40%