Trump Defense 2020 and Monte-Carlo tree search

# Jernej Habjan

12. December 2017

# Commonly used expressions:

|  |  |
| --- | --- |
| Actor | Any object that can be placed into level |

# Trump Defense 2020:



## Game State and Game Instance

»GameInstance is a class that which state persists switching of levels, game mode switches etc. where classes like GameMode or PlayerController are being reset and any data stored in them is deleted and put to defaults again. Any data that you want to keep beyond levels and matches, for example "what player did in specific moment so you can have consequence on other level" should be placed here. This class is mainly helpful for single player games... but it can find uses in multi player too ;] I not sure if that class is replicated« – Shadowriver from Unreal Engine forum

### Saving game state:

Game state can be saved in variables such as arrays or dictionaries where key is each player.

For each player game state consists of:

* Buildings
* Units
* Resources collected (Money)

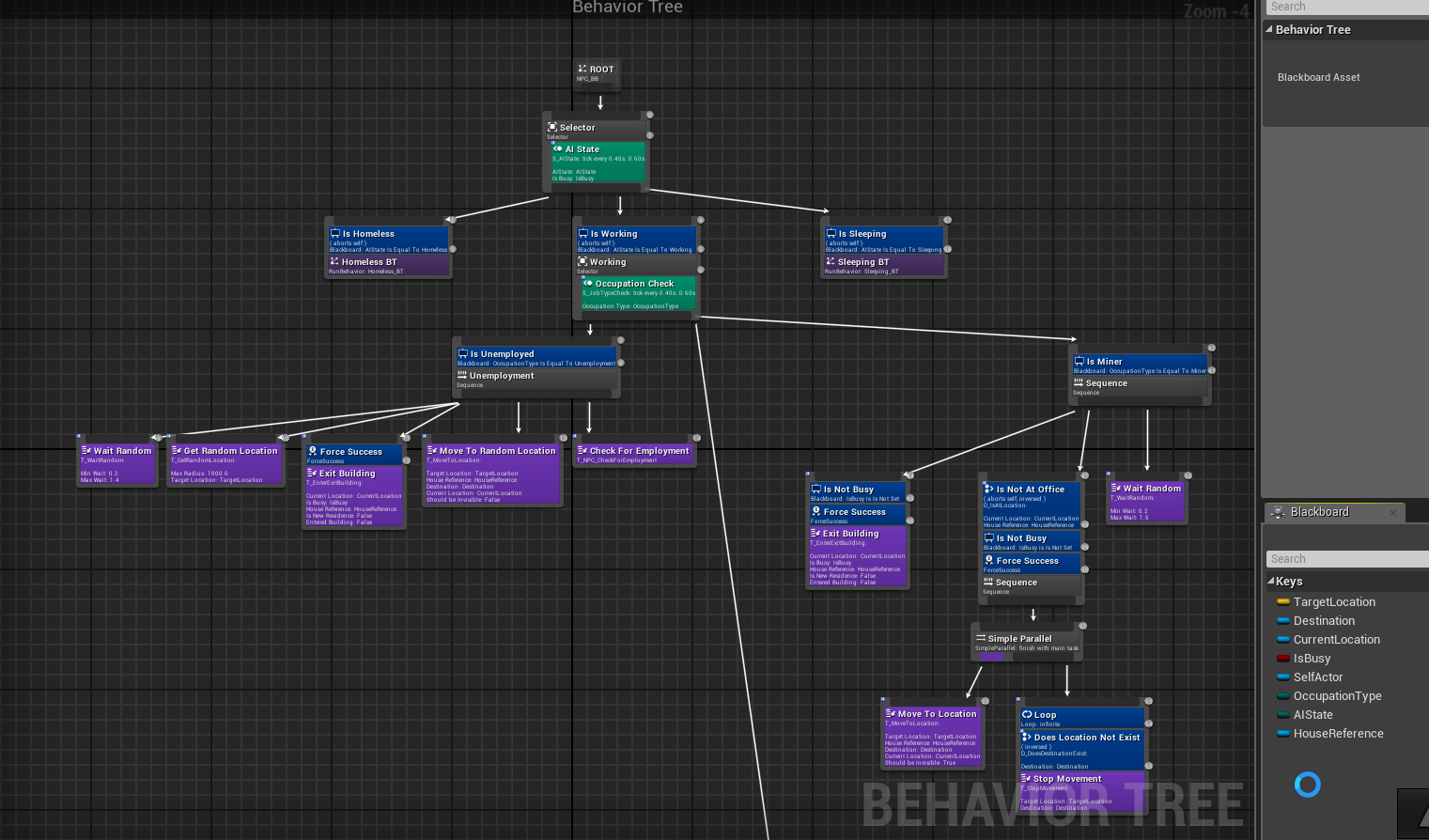
1. Buildings:
   1. Health
   2. Construction progression
   3. Occupants / workers
   4. Unit training progression
   5. Position
2. NPCS:
   1. Health
   2. Position
   3. Action / Blackboard keys
3. Resources (for both players same):
   1. Resources remaining

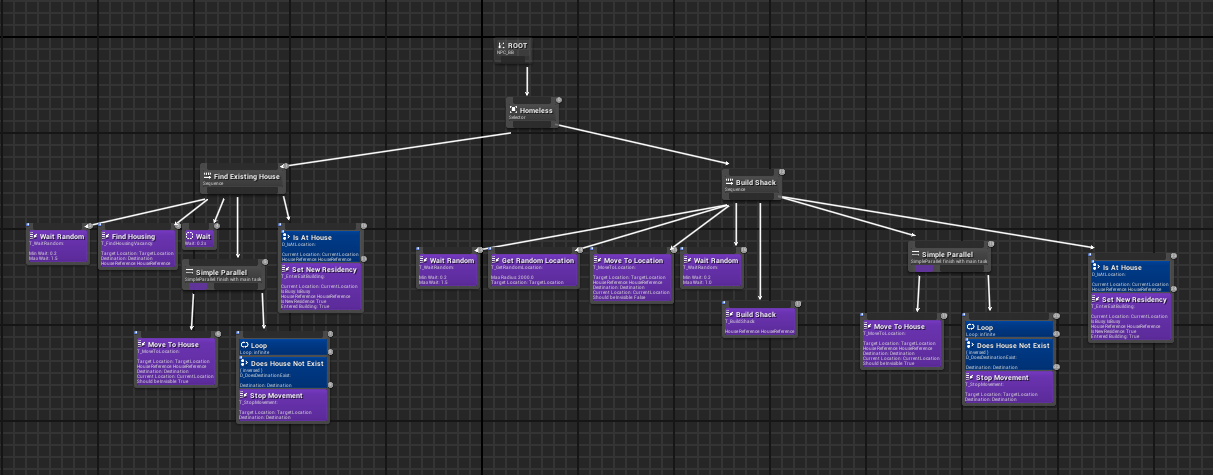
## Speeding up game for simulation

Event tick is used to update ingame time, AI recieves TickExecute event to check for AI state change and when creating building, tick is used to update build time.

That is too slow. Have to run trees manually without tick and finish them with stop logic (personal note).

## Behavior Trees





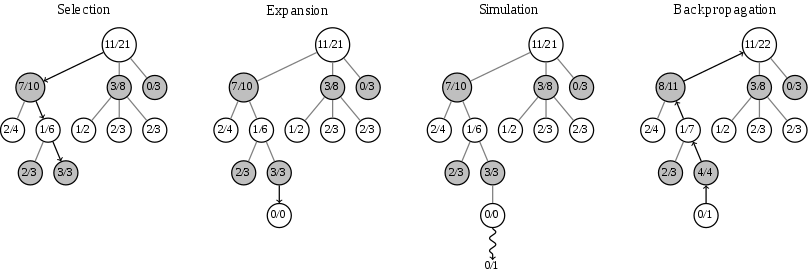
### Behavior trees

* NPC miner
* NPC construction
* NPC homeless
* NPC idle
* NPC attack actor

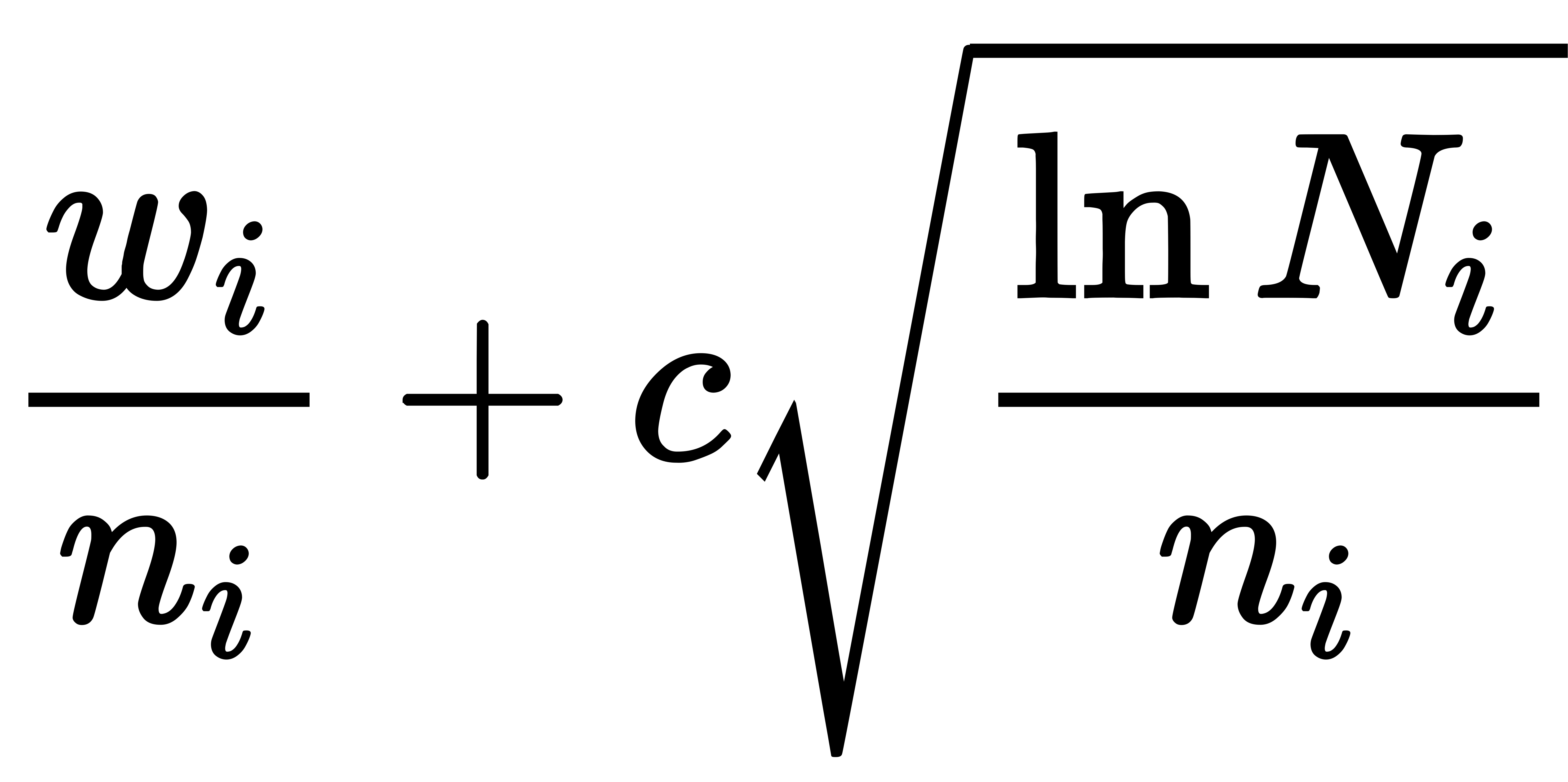
### Actions

* Construct residence
* Construct construction office
* Construct mining shack
* Construct barracks
* Create marine
* Create worker

## Monte Carlo Tree Search



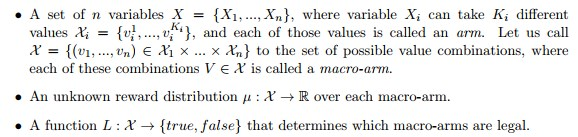
### UCB1



### RAVE

Using win rates of already explored nodes.

### Naive MCTS

It is supposed to be most appropriate for RTS games, as it's working with big combinatorial spaces and it assumes that awards are independent of each other. 

### Determining win/lose

Ratio between your and your opponents score.  
Score is calculated as number of units \* their health, number of buildings \* their health.

## Applicating algorithm on multiple actors in same tick for N players.

Multiple NPCS, buildings and players.

## Next step

Reinforcement learning?