3D Path Finder where gravity works

Decision Transformer

Introduction members

2015004984 정동훈

Computer Software Dept.

2014004211 표영권

Computer Software Dept.

Idea from

Perseverance, observing Mars robot, by NASA

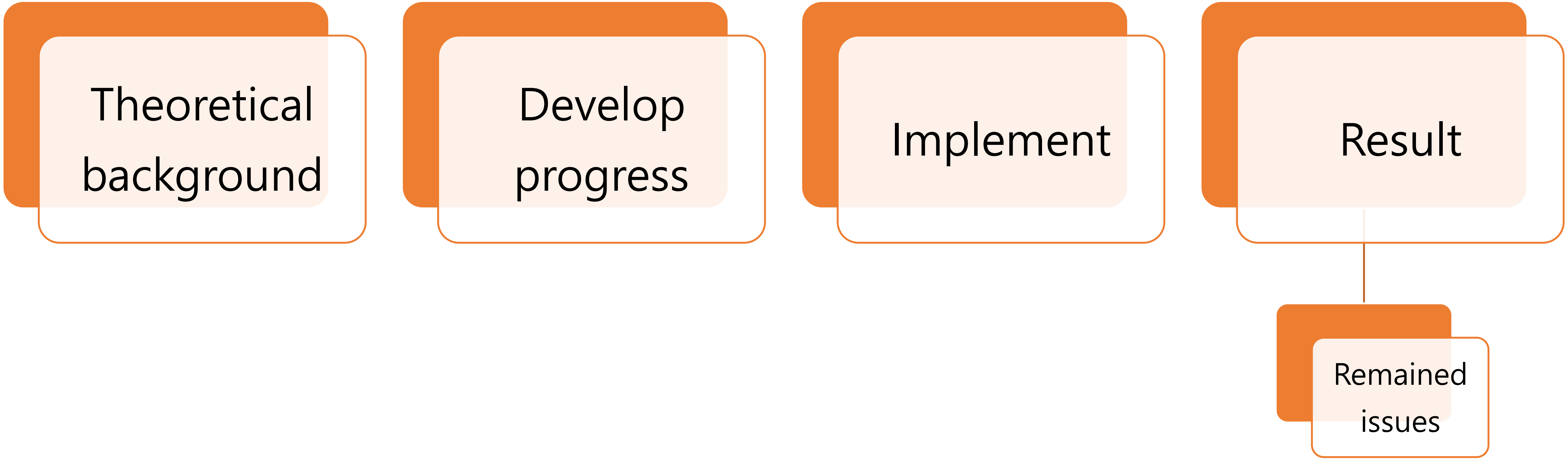
Object

- Making path finder on 3-Dimensional space

- Constraints:
  - **Gravity** works with $9.8m/s^2$
  - Agent has own **stamina** and **HP** so that something can happen if the attributes were starved.
  - Agent behaviors run, splint, jump, climbing, and gliding(use parachute). Agent can also just wait.

- Finding paths where the shortest time would be spent.

Contents



Theoretical background

Develop progress

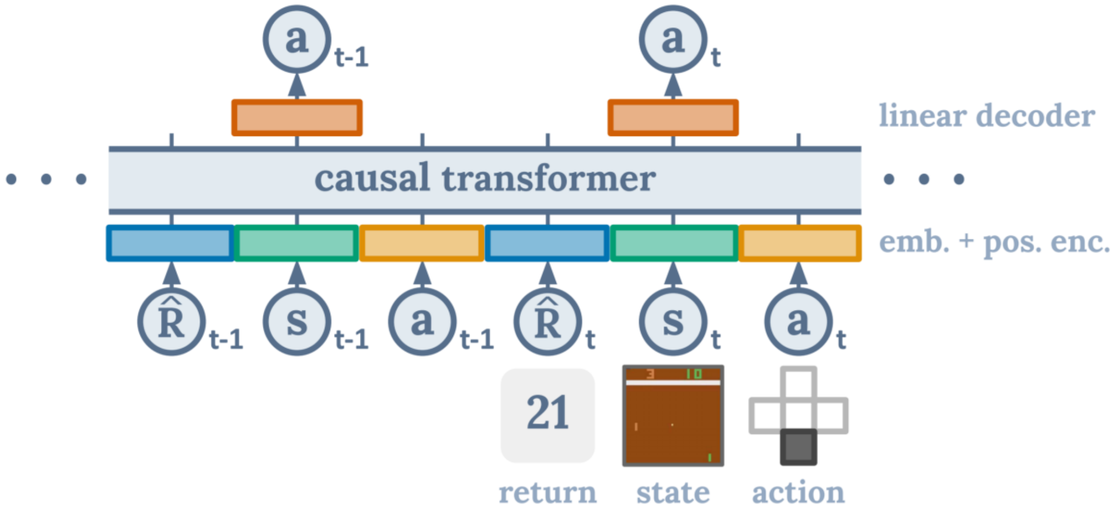
Implement

Result

Remained issues

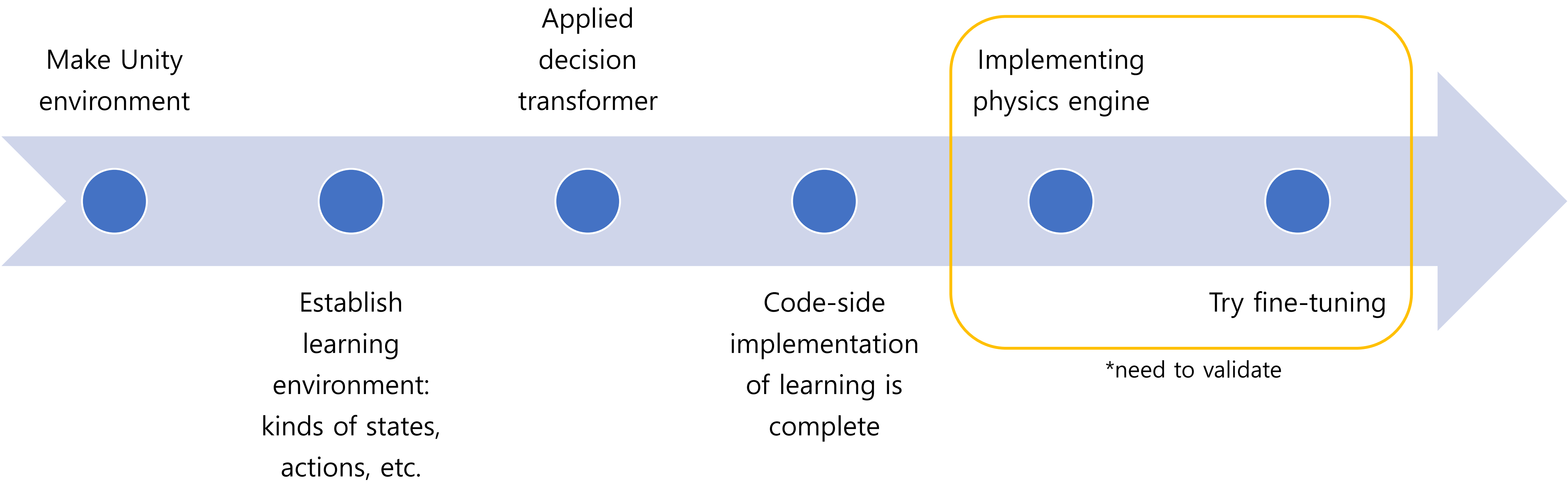
Theoretical background

- ***Decision Transformer**: Reinforcement Learning via Sequence Modeling
  - **Model-Free** : Just use record tuple sequence!
  - Records for reinforcement learning are given as sequence of tuples, (r_t, s_t, a_t, t)
  - 'return' is obtained by sum of expected results in the future from timestep t → $\widehat{R_t} = \sum_{k=t}^{T} r_k$



Decision Transforemer - Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, Igor Mordatch, Jun 2021

Develop progress



Make Unity environment

Establish learning environment: kinds of states, actions, etc.

Applied decision transformer

Code-side implementation of learning is complete

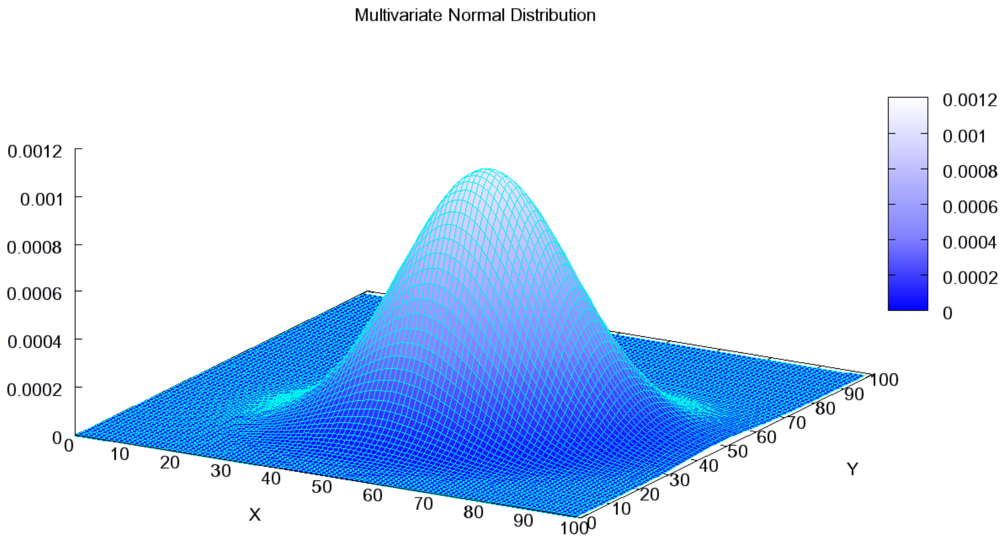
Implementing physics engine

Try fine-tuning

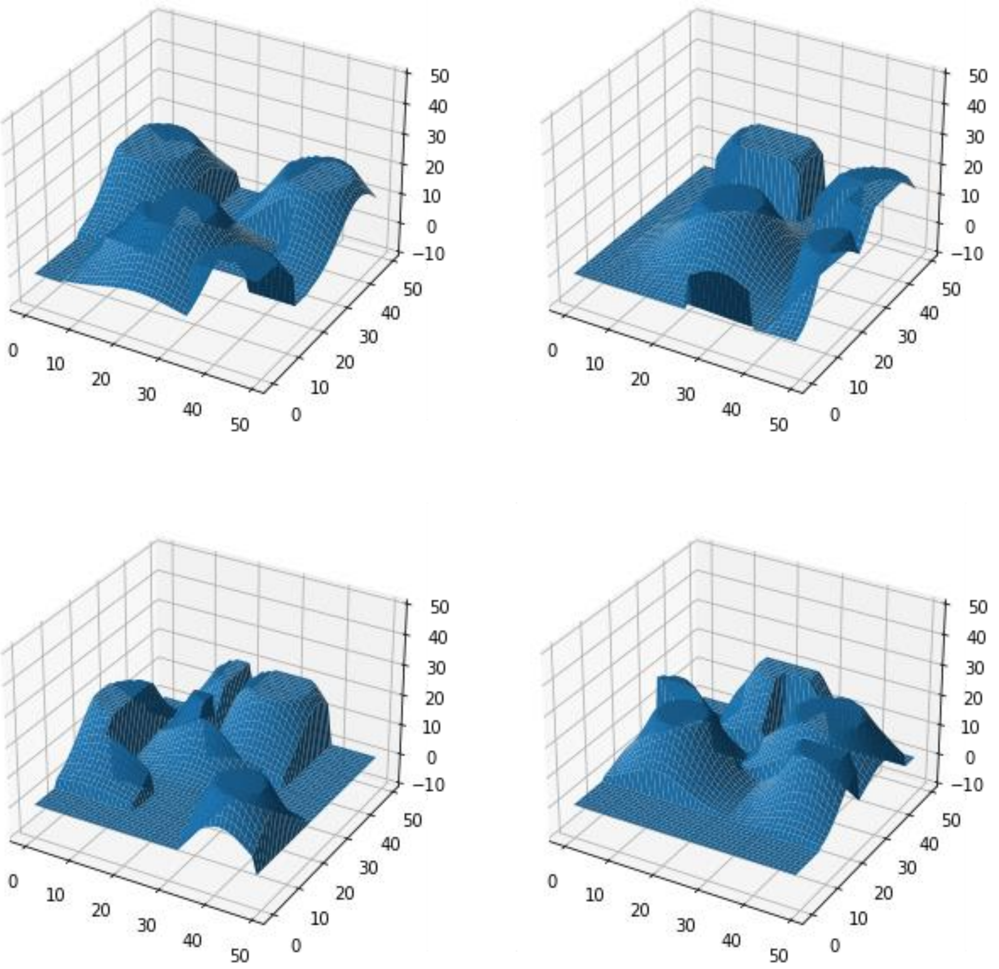
*need to validate

Implement

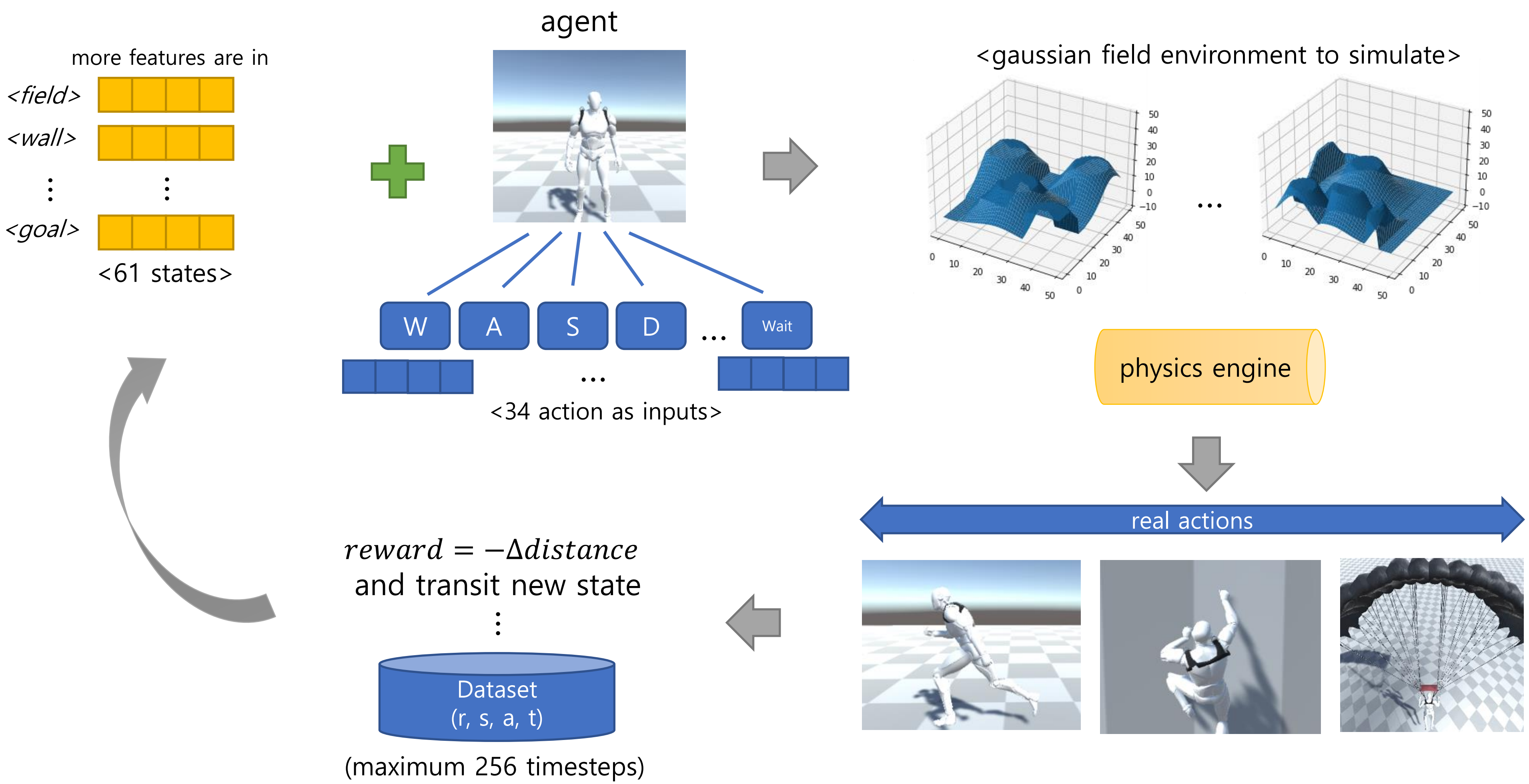
Map designer



Width	Height	Depth	n	# of maps
50	50	20	5	12



Design of learning environment



agent

more features are in

<field>

<wall>

⋮

<goal>

<61 states>

W A S D … Wait

…

<34 action as inputs>

<gaussian field environment to simulate>

…

physics engine

real actions

$reward = -\Delta distance$

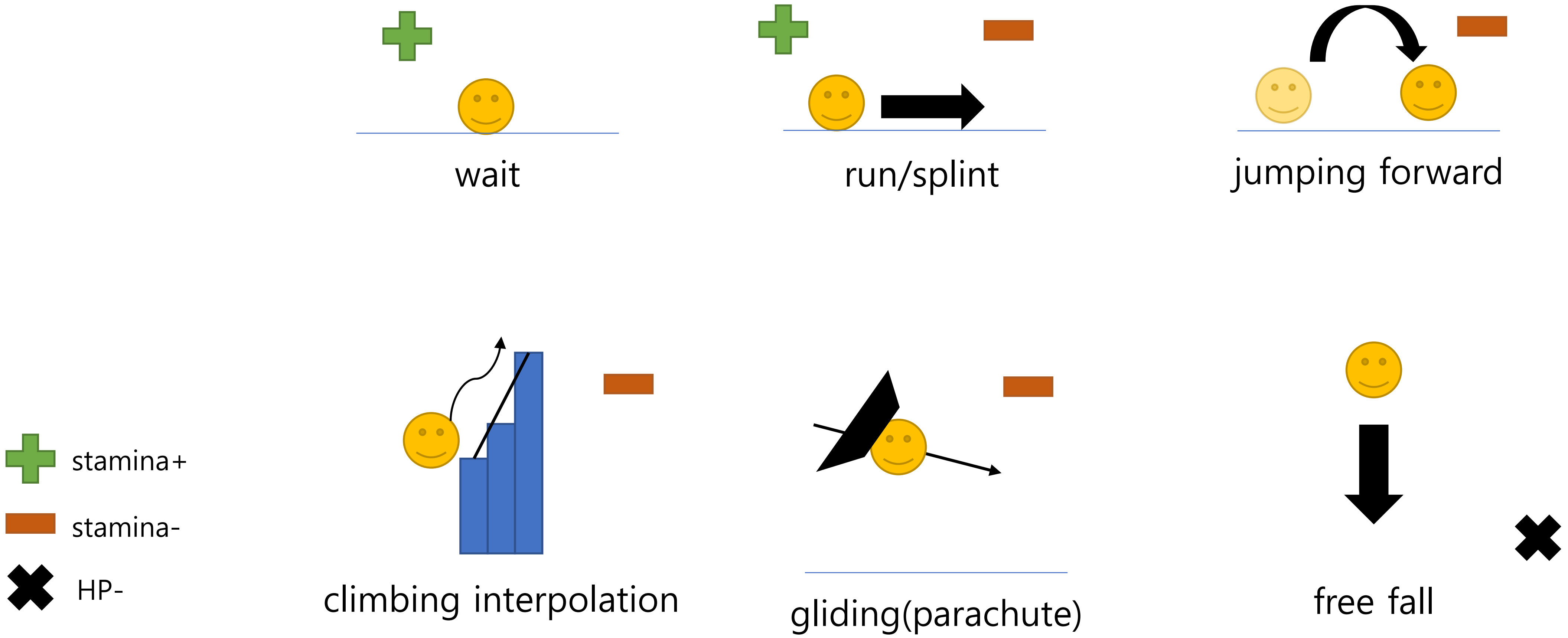
and transit new state

⋮

Dataset
(r, s, a, t)

(maximum 256 timesteps)

Physics engine



wait

run/splint

jumping forward

climbing interpolation

gliding(parachute)

free fall

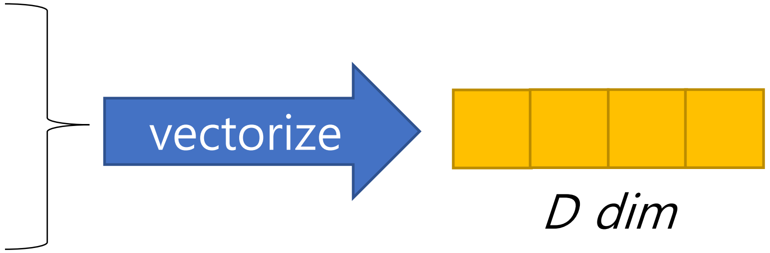
✚ stamina+

▬ stamina-

✖ HP-

Others

- agent.py – representing several attributes of agent.
    - HP, stamina, viewing direction vector, present position vector, doing action
- action.py – representing motion of each action.
    - id, input key, velocity vector, stamina consume info
- state.py – representing the state of the agent.
    - id, no, remained distance, spend time



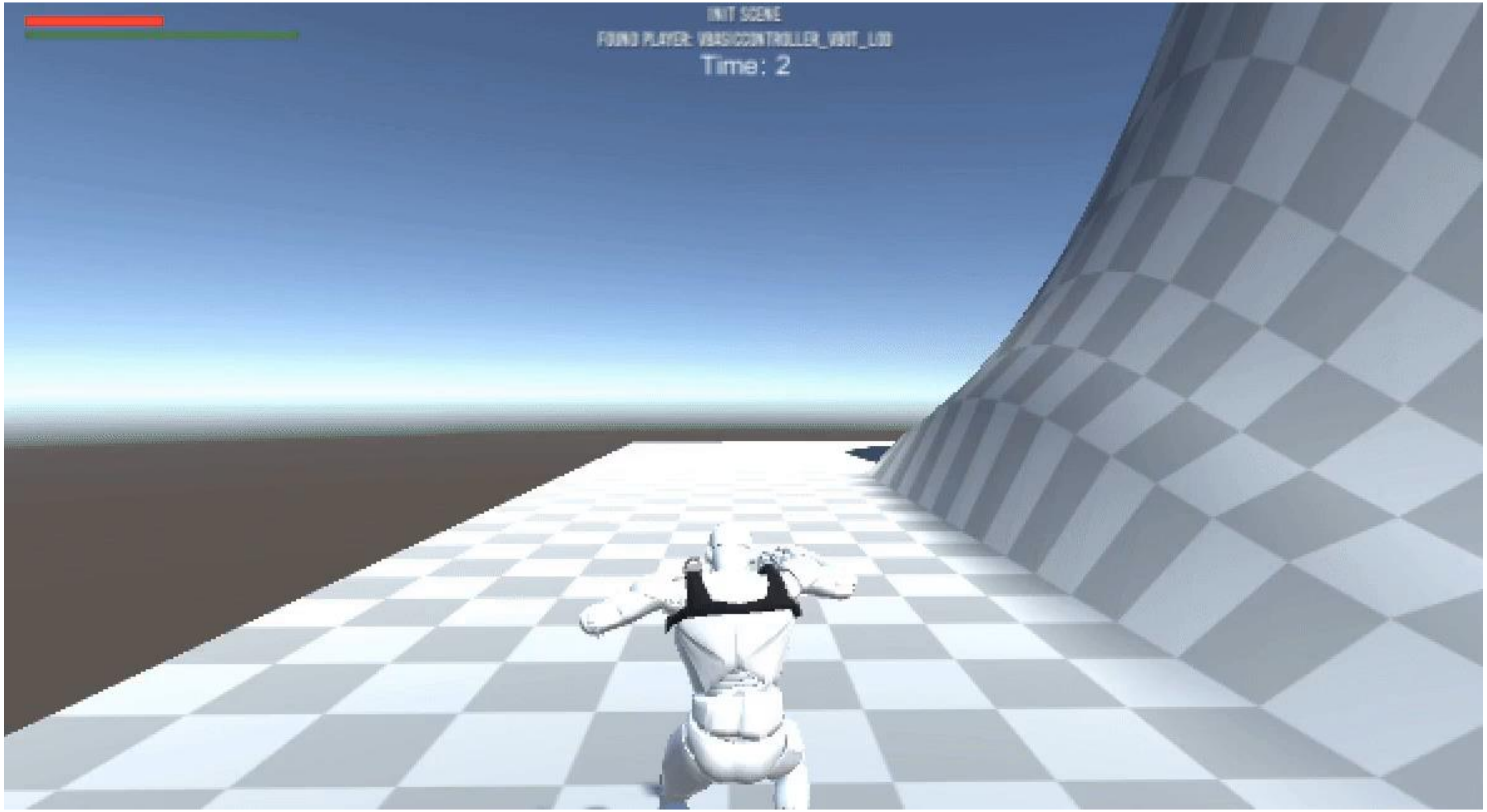
vectorize

D dim

- basic_math.py – representing vector operations
    - rotation, projection, size, normalize, distance

Result

Traversal example



https://youtu.be/irt4aDOwPtI

Remained Issues

- Improve the strategy of reward
- Multiprocessing upgrade
  - each child process traverses each environment
  - the speed of generation will be faster if they do job in same environment
- Fine tuning portion of the learning
  - Hyper parameters we referred were not fit our model

Thank you