**Developing snake using Reinforcement learning**

**Game (Pygame)**

* play\_step(action)
  + reward, game\_over, score

*(Play step gets an action and then does a step. After the move, it returns the current reward, if the game is over or not and also the current reward)*

**Agent**

* game
* model

We store both the game and model in our agent

Training:

* state = get\_state(game) *(Based on the game we have to calculate the state)*
* action = get\_move(state):
  + model.predict() *(Based on the state we calculate the next action, and this involves calling model.predict())*
* reward, game\_over, score = game.play\_step(action) *(From the action, we call play\_step() where we get the reward, game\_over state and the score)*
* new\_state = get\_state(game) *(With the above information, we calculate the new\_state)*
* remember *(Using the old\_state, new\_state, game\_over state and score we remember all of these information)*
* model.train() *(With all of the above, we train our model)*

**Model (PyTorch)**

* Linear\_QNet(DQN)
  + Model.predict(state)
    - * Action

*(Linear\_QNet is a feet forward neural network with a few linear layers and need information such as new\_state and the old state and we can call model.predict() which gives us the next action)*

Reward

* Eat food: +10
* Game over: -10
* Else: 0

Action

[1, 0, 0] -> straight

[0, 1, 0] -> right turn

[0, 0, 1] -> left turn

State (11 values)

[danger straight, danger right, danger left,

direction left, direction right, direction up, direction down,

food left, food right, food up, food down]

A diagram of a model

Description automatically generated

(Deep) Q Learning

Q Value = Quality of action

1. Init Q value (= init model)
2. Choose action (model.predict(state)) (or random move) *(Sometimes choose a random moves at the beginning when we don’t know a lot about the game)*
3. Perform action
4. Measure reward
5. Update Q Value (+ train model)
6. Repeat from 1

Q Updated Rule Simplified:

A close-up of a math equation

Description automatically generated with medium confidence

Q is model predict with the old state and then the new state is reward plus the gamma value multiplied by the maximum Q value of state 1

Loss function:

A black arrow pointing to a black curve

Description automatically generated