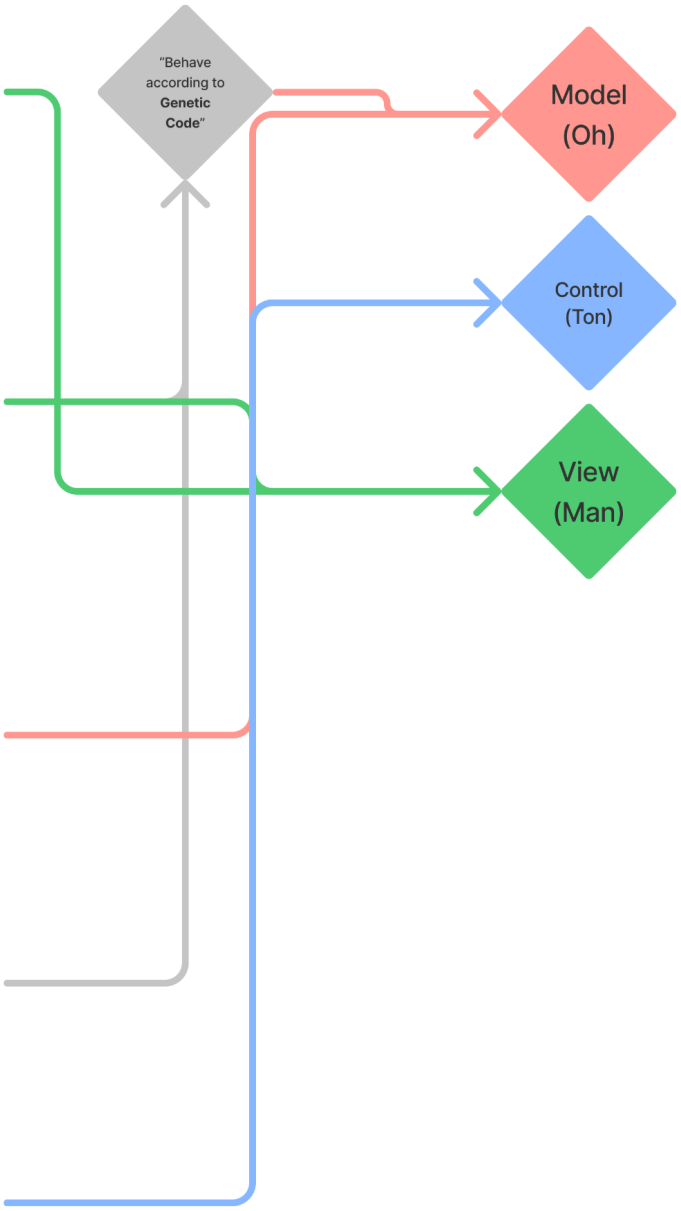
FrontPage:

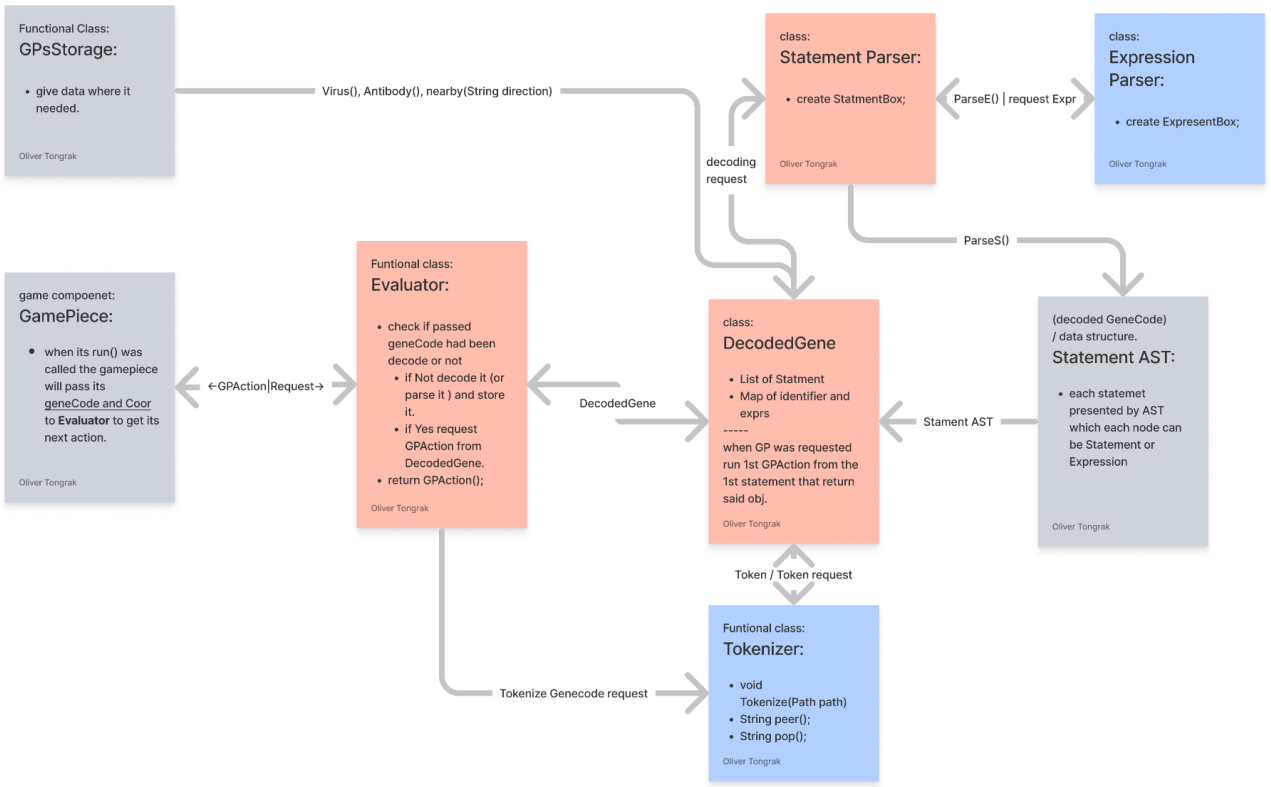
CPE200: Proj: CARIN:
Group: YukonGold:
Genetic Code Evaluator Report:



Genetic Code Evaluator Report:

Architecture:

Classes relationship diagram:

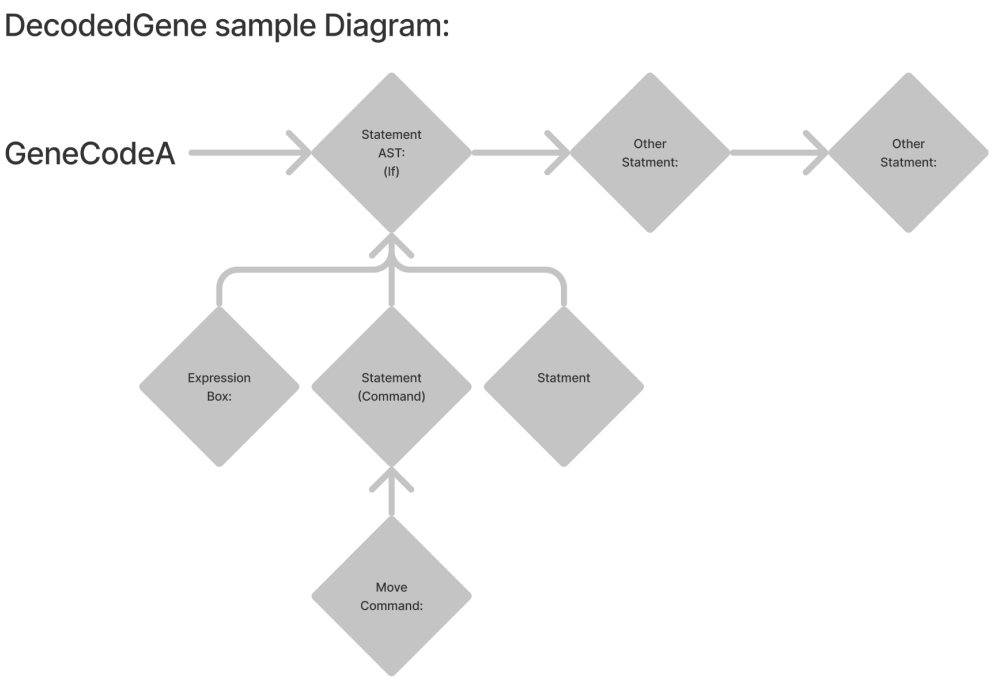


(Figure 1: Classes Relationship Diagram)

Important class and components:

The proj spec requires that all action of all game pieces (both virus and antibody) must come from a predetermined Genecode, Which has to be decoded (through Evaluator alike) into the action which will become that game piece's next action. Our group had designed a decoding step to work a little differently than said statement. As we will see in Figure1.

The evaluator class will call for genecode decoding if it receives a new genecode. Then the evaluator will call for a creation of a new DecodedGene object, which will be a representation of a new kind of genetic code. Each DecodedGene stores its gene information in a form of list of AST Statements. List of AST Statements means each statement contains another AST Component as its type needed. For example in Figure2.



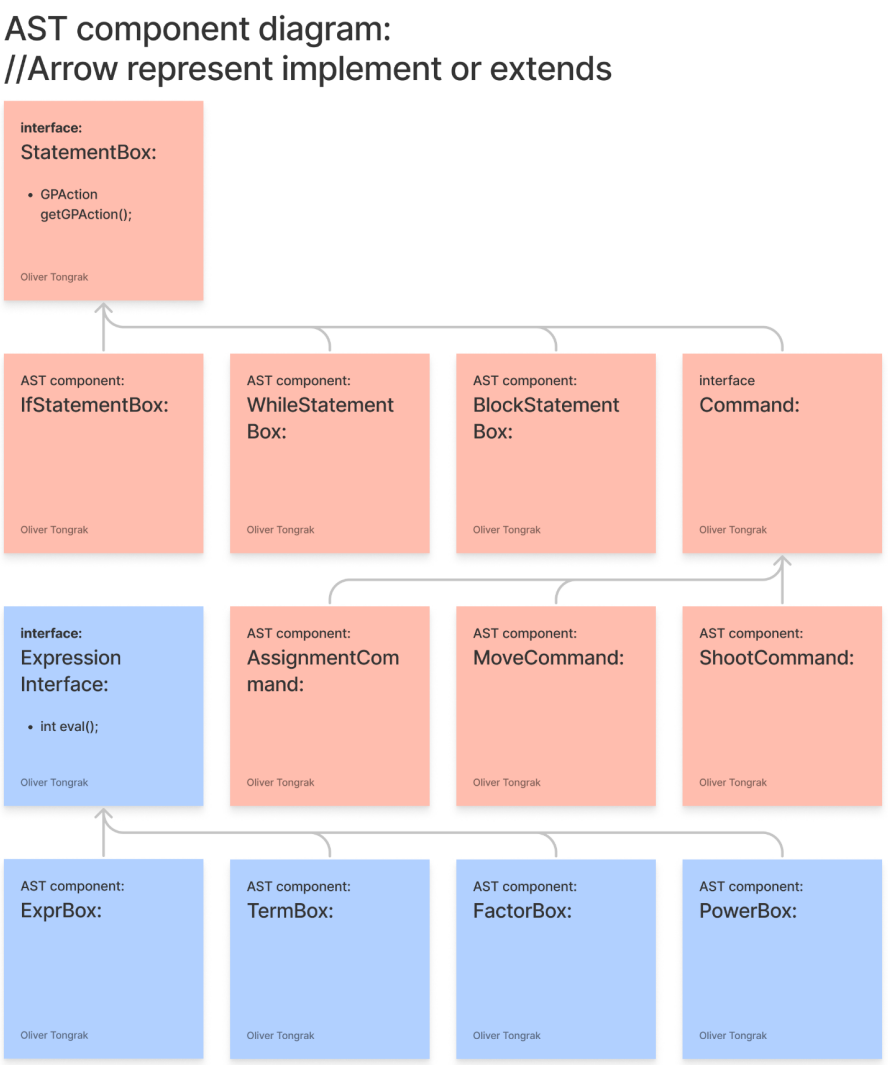
(Figure 2: DecodedGene sample Diagram)

In another case(if evaluator receives familiar GeneticCode), evaluator will send client's Coor and request client's next action from DecodedGene. DecodedGene formed the client's action by using their Coor to "work-out" each statement in its list and if a statement returns the client's action, pass the action instruction to client.

To "work-out" a statement means that a statement has to return its GPsAction which if the statement is a nested statement like:If, While, and Bracket, it has to get its GPsAction from appropriate nested statements. This process will continue until it reaches a CommandStatement. Then if it is a Move or Shoot command its action instruction has to be returned to the client. Else (said command statement is Assignment) the action instruction will be null. Which the DecodedCode will take it as AssignementStatement and have to get client's action form next statement instance.

the creation of new DecodedGen iff new genecode was detected, ensuring that we can reuse pre-decoded GeneticCode with the same type of game piece in the future.

Regarding the AST statement components, each component represents a decoded genetic code. All AST's components are as Figure3. Each AST's component was designed according to the given genetic code, but some part of the grammar was simplified like: ActionCommand was dissolved into MoveCommand, and ShootCommand.



(Figure3: AST component diagram)

What we had learn:

- Creating a class/ component without a proper plan is a bad idea.
- Using interface and abstract classes can make code implementation goals clearer and easier to integrate with others.

Genetic Code Evaluator Report:

Testing:

Testing Description:

We have tested in the following classes:

- Tokenizer → The tokenizer can split tokens correctly?
- StatementParser → able to manage That statement can be correct?
- ExpressionParser → Can parse expressions correctly?
- GeneEvaluator → can the genetic code be obtained and analysed and it is possible to obtain a new genetic code that can be analysed if the received genetic code has been analysed can it be reused?
- DecodedGene → Able to manage a list of statements?

Test Cases:

Tokenizer: Create a new GeneticCode file using a sample from the professor as a reference and try running the program to test if the token stream is correct. For example

```
virusLoc = virus
if (virusLoc / 10 - 1)
then
  if (virusLoc % 10 - 7) then move upleft
  else if (virusLoc % 10 - 6) then move upright
  else if (virusLoc % 10 - 5) then move left
  else if (virusLoc % 10 - 4) then move right
  else if (virusLoc % 10 - 3) then move downleft
  else if (virusLoc % 10 - 2) then move downright
  else if (virusLoc % 10 - 1) then move down
  else move up
```

StatementParser : See how that statement works.
For example:

```
if (virusLoc % 10 - 7) then move upleft
```

It is able to work according to the state it is sent in.

ExpressionParser :It is able to parse the expression correctly.
For example:

```
if (virusLoc % 10 - 7) then move upleft
```

It can be argued that `virusLoc` is identifier.

GeneEvaluator :

```
if (virusLoc % 10 - 7) then move upleft
```

DecodeedGene :

What we had learn:

- implementing without testing a single thing is a free ticket to “Integration hell”.
- Do learn things that you used to know but have forgotten.

Genetic Code Evaluator Report:

Work Plan:

Roles Description:

As you might notice, in the above diagrams each post-it was colour differently. Each colour represents a separation of workloads in geneCode Evaluation implementation. Another fact is each colour actually represents each component in MVC idea (as represented in the front cover diagram). The main reason why said colour coordination was used in geneCode Evaluation implement as well, was the person responsive in Model field requires help from View and Controller field.

What we had learn:

- A clear interface is the best tool in a group project.

Genetic Code Evaluator Report:

Knows Problem:

Due to ill-planning, our group suffers a little thing called “behind schedule pressure”. Meaning, the implemented genecode evaluator, although seen finished, hadn’t been testing properly. This signifies to us that the genecode evaluator **couldn't be considered finished**. And it needs more testing and fixing with more time than what had been given. The estimated extra time needed for both testing and fixing is at most 24 hours.