Object-Oriented Programming
Fall 2015

Project 5

Seon Joo Kim

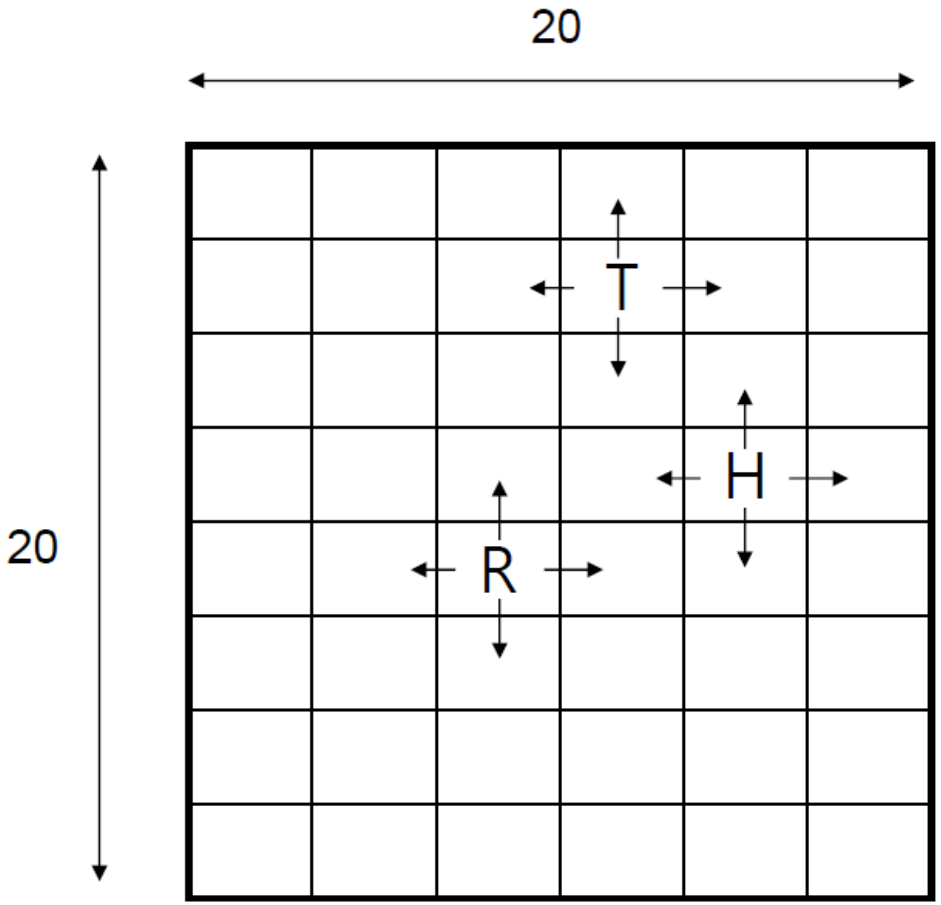
Outline

- Project Overview
- Programming Problems
- Requirements & Assumptions
- Deliverables, due-date and submission

Project Overview

- Simulation of survival game.
  - Character
    - Tiger
    - Hunter
    - Rabbit
    - Grass
    - Pit
- Word
  - 20x20 grid cells
  - One critter only can occupy a cell at a time
  - Critter can’t get out of given world

Simulation Space



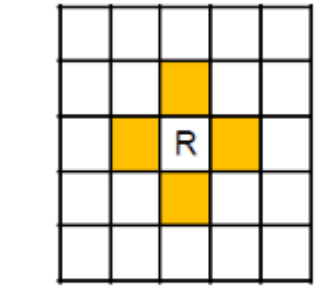
- T : Tiger
- H : Hunter
- R : Rabbit

- O : pit
- * : Grass

Outline

- Project Overview ✓
- Programming Problems
- Requirements & Assumptions
- Deliverables, due-date and submission

Rabbit's Action

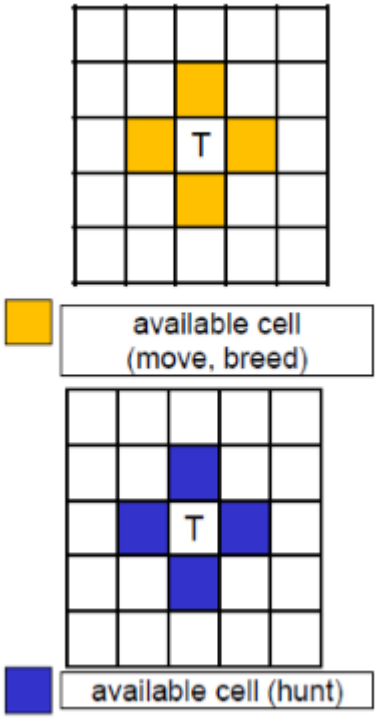


- Move
  - At every time step, move randomly
    - Up, down, left, right
    - If the neighboring cell in selected direction is occupied or off the grid
      - Stay in the current cell
  - If there are grasses in adjacent cells
    - A rabbit will move to one of those cells and eat a grass
- Breed
  - When a rabbit survives for three times, then it breed a child at the end of the time step(that is; after moving)
    - Creates a new rabbit in an adjacent cell which is empty
    - No empty cell, no breeding occurs

Rabbit's Action

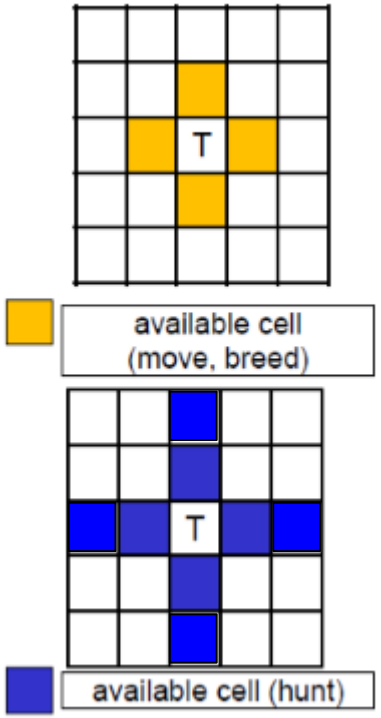
- Starve
  - If a rabbit has not eaten a grass within the last three time steps, the rabbit will die after moving

Tiger's Action



- Move
  - If there are rabbits in adjacent cells
    - A tiger will move to one of those cells and eat a rabbit
    - Assume that tigers do not eat each other
    - Tiger can't attack hunter
  - Other rules are same as rabbit's
- Starve
  - If a tiger has not eaten a rabbit within the last three time steps, the tiger will die after moving.
- Breed
  - If a tiger survives for four time steps
    - Spawn off a new tiger in the same manner as rabbits

Hunter's Action



- Move
  - If there are rabbits or tigers in up to 2 distance cells
    - A Hunter will move one of those cells and eat a critter
    - Hunter do not attack each other
    - Hunter can move 2 cells and kill if there are some critters
  - Other rules are same as rabbit's
- Starve
  - If a hunter has not eaten a rabbit or a tiger within the last three time steps, the hunter will die after moving
- Breed
  - If a hunter survives for eight time steps
    - Spawn off a new hunter un the same manner as rabbits

Grass and Pit

- No move, No starve, No breed
- If the grass was eaten, it will grow up in another cell that is not occupied
  - If the grass was stepped by critters, it will do same
  - The number of grasses will be maintain
- If a critter fall into the pit, it will die
  - A critter dose not know where pits are
  - If a critter spawn off a new critter into a pit , it will die

Outline

- Project Overview ✓
- Programming Problems ✓
- Requirements & Assumptions
- Deliverables, due-date and submission

Requirements & Assumptions

- At the beginning, take the initial number of rabbits, tigers and hunters from user input.
  - Exception handling is not required.
  - Of course, we assume that input never exceed 4000. (20x20)

- Draw 2-dimensional world
  - ‘R’ for rabbits
  - ‘T’ for tigers
  - ‘H’ for hunters
  - ‘O’ for pits
  - ‘*’ for grasses
  - ‘-’ for empty cell
  - Use a 2-dimensional array (e.q., world[20][20])

Requirements & Assumptions

- This time, please follow exact output format asked.
  - see the execution example
- Print current time at the top of the screen.
- Print event message at the bottom of the screen how many critters born, starve, hunted at that step.
- After each time step, prompt the user to press 'Enter' to move to the next time step and clear screen.
  - Hint: getchar(); system("clear");
- Movement priorities: Hunter > tiger > rabbit.
  - ex) Only survived rabbit after tigers' movement can move.

Requirements & Assumptions

- Use Classes for each characters(Rabbit, Tiger, Hunter).
  - 3 member variables: indicate **x & y coordinates** and **lifetime** each
  - 2 member functions: **move**(), **breed**()
  - For breeding or starving, you must use new, delete keyword.
  - Allocating 400 objects for each critter at the beginning is not allowed.
  - You can create additional classes, member variables, member functions if you need
  - You can use ‘rand’ function for moving
- Use reasonable class feature (private/public, static, const, constructor)
- Reasonable comments in English(Important!)
- Reasonable Indentation(Important!)

Execution Example (for simplicity, 8x8 area)

- Initialization

```
Enter initial number of rabbits : 5
Enter initial number of tigers : 2
Enter initial number of hunters : 2
Enter initial number of grasses : 5
Enter initial number of pits : 2
```

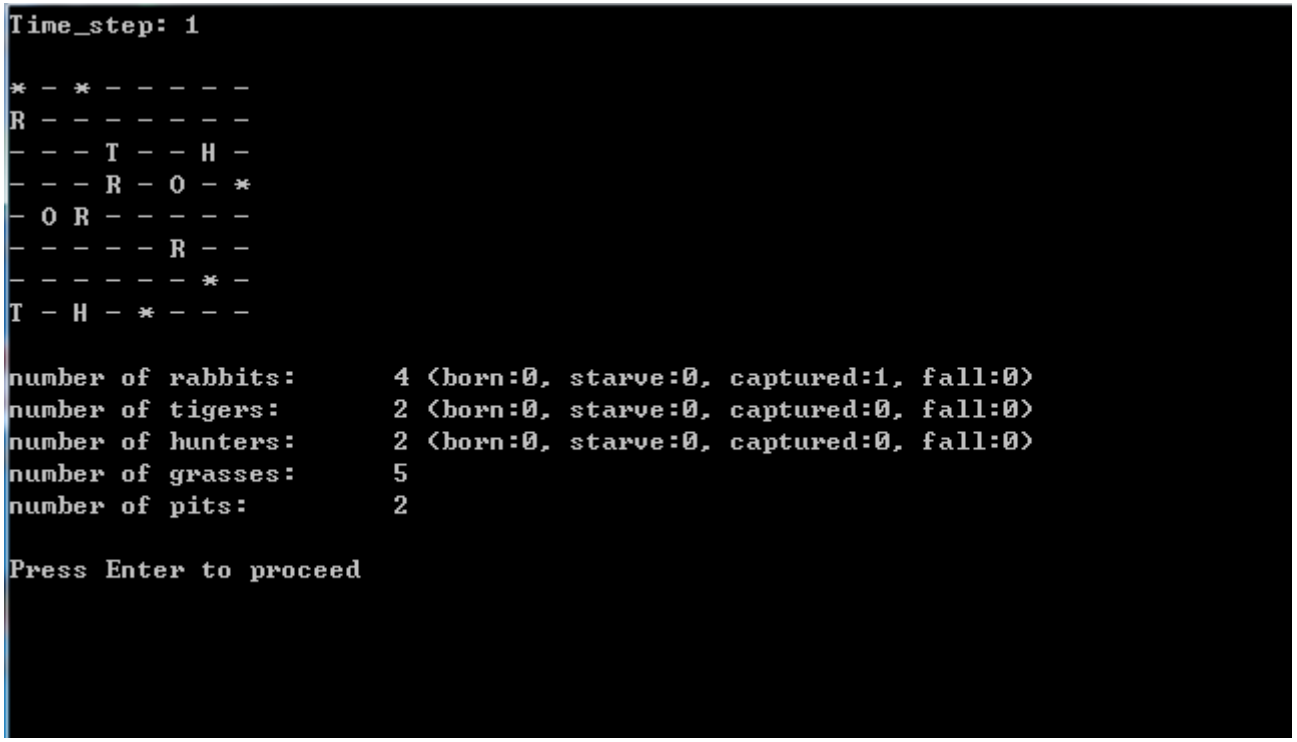
Execution Example (for simplicity, 8x8 area)

- step 0

```
Time_step: 0

* - * - - - - -
- R - - - - - -
- - - * T - R -
- - R - - O - *
- O - - - - H -
- - R - - - - -
T - - - - R - -
- H - - * - - -

number of rabbits:      5 (born:5, starve:0, captured:0, fall:0)
number of tigers:       2 (born:2, starve:0, captured:0, fall:0)
number of hunters:      2 (born:2, starve:0, captured:0, fall:0)
number of grasses:      5
number of pits:         2

Press Enter to proceed
```

Execution Example (for simplicity, 8x8 area)

- step 1



Execution Example (for simplicity, 8x8 area)

- step 2

```
Time_step: 2

R - * - - - - -
- - - - - - - -
- - - - - - - -
- - - T - O H *
- O - - - - - -
- - - - - - - -
- * - - - R * -
H - - - * - - -

number of rabbits:      2 (born:0, starve:0, captured:1, fall:1)
number of tigers:       1 (born:0, starve:0, captured:1, fall:0)
number of hunters:      2 (born:0, starve:0, captured:0, fall:0)
number of grasses:      5
number of pits:         2

Press Enter to proceed
```

Execution Example (for simplicity, 8x8 area)

- step 3

```
Time_step: 3

- - * - - - - -
R - - - - - - -
R - - - - - - -
- - - - T O - *
- O - - - - H -
- - - * - - R -
H * - - - - R -
- - - - * - - -

number of rabbits:      4 (born:2, starve:0, captured:0, fall:0)
number of tigers:       1 (born:0, starve:0, captured:0, fall:0)
number of hunters:      2 (born:0, starve:0, captured:0, fall:0)
number of grasses:      5
number of pits:         2

Press Enter to proceed
```

Execution Example (for simplicity, 8x8 area)

- step 4

```
Time_step: 4

R - * - - - - -
- - - - - - - -
- - - - T - - -
R - - - T O - *
- O - - - - - -
- - - * - - H -
- * - - - R - -
H - - - * - - -

number of rabbits:      3 (born:0, starve:0, captured:1, fall:0)
number of tigers:       2 (born:1, starve:0, captured:0, fall:0)
number of hunters:      2 (born:0, starve:0, captured:0, fall:0)
number of grasses:      5
number of pits:         2

Press Enter to proceed
```

Execution Example (for simplicity, 8x8 area)

- step 5

```
Time_step: 5

- - * - - - - -
- - - - - - - -
- - - - - - - -
- R - T - O - *
- O - - - - - -
- - - * - - - -
- * - - R - H -
- - - - * - - -

number of rabbits:      2 (born:0, starve:1, captured:0, fall:0)
number of tigers:       1 (born:0, starve:1, captured:0, fall:0)
number of hunters:      1 (born:0, starve:1, captured:0, fall:0)
number of grasses:      5
number of pits:         2

Press Enter to proceed
```

Outline

- Project Overview ✓
- Programming Problems ✓
- Requirements & Assumptions ✓
- Deliverables, due-date and submission

Marking Criteria and Plagiarism

- **Marking Criteria**
  - Score is only given to programs that compile and produce the correct output.
  - Points are deducted for programs that produce compiler warnings. Hint: use the –Wall command-line parameter to eliminate all warnings.
  - Points deductions on programming style: provide comments in your code and use proper indentation of lines.
  - Please pay particular attention to the requested **output format** of your programs. Deviating from the requested output format results in points deductions.
  - Program should work correctly in our **VM**.

- **Plagiarism (Cheating)**
  - All submissions are checked for plagiarism.
  - Once detected, no score will be given for the lab to all students involved in the plagiarism incident.

Deliverables

- This time, your are required to separate one source file into several headers and source files. (One header and source for each class)

ex) project5.cpp →
rabbit.h rabbit.cpp
tiger.h tiger.cpp
hunter.h hunter.cpp
grass.h grass.cpp
pit.h pit.h
(+ other additional class you introduce)
(+ header for shared variable if need) ex) common.h
project5.cpp

- CLASSNAME.h: contain class declarations
- CLASSNAME.cpp: contain class definitions
- project5.cpp: contain main functions

Deliverables

- How to compile:

(Assume we have 11 files - rabbit.h rabbit.cpp tiger.h tiger.cpp hunter.h hunter.cpp pit.h pit.cpp grass.h, grass.cpp project5.cpp)

1. g++ -c rabbit.cpp tiger.cpp hunter.cpp grass.cpp pit.cpp
2. g++ -o output project4.cpp rabbit.o tiger.o hunter.o pit.o grass.o

OR

1-2. g++ -o output rabbit.cpp tiger.cpp hunter.cpp grass.cpp pit.cpp project5.cpp

3. ./output (will execute the program)

Deliverables

- File list to submit
  - CLASSNAME.h, CLASSNAME.cpp (for each class)
  - **project5.cpp**

Warning: you will lose points if the file name is not proper!

Archiving the deliverables

- Please zip the all files of your submission to a single archive:

  $ tar –jcvf **project5_<student_id>.tbz2** ALL_HEADERS.h ALL_SOURCES.cpp project3.cpp

- To make sure, you can extract the archive file with following command:

  $ tar –jxvf **project5_<student_id>.tbz2**

- Please note that in the above command, all must be typed in on a single line!
  - The shell will wrap-around the text for you ☺

Submitting your archive

- You are asked to upload your archive on YSCEC.
  - **project5_<student_id>.tbz2**
- Due date: Nov. 16, 23:55PM.
- For instructions on how to upload a file on YSCEC, please see Lecture Note 2,

Outline

- Project Overview ✓
- Programming Problems ✓
- Requirements & Assumptions ✓
- Deliverables, due-date and submission ✓