Software Modeling and Design

Tongji University - Xavier Ferré

Hello!

ARNAUD ZHENG

21years old
Tongji - Epitech Paris
“What does the snake says?”

Hello!

GABRIEL TANG

22 years old
Tongji - Epitech Paris
“I ate an apple”

Hello!

STEPHANE KHAU

21years old
Tongji - Epitech Paris
“Hiss Hiss~”

Summary

1 THE GAME

2. CLASS DIAGRAM

3. SEQUENCE DIAGRAMS

4. DESIGN PATTERN

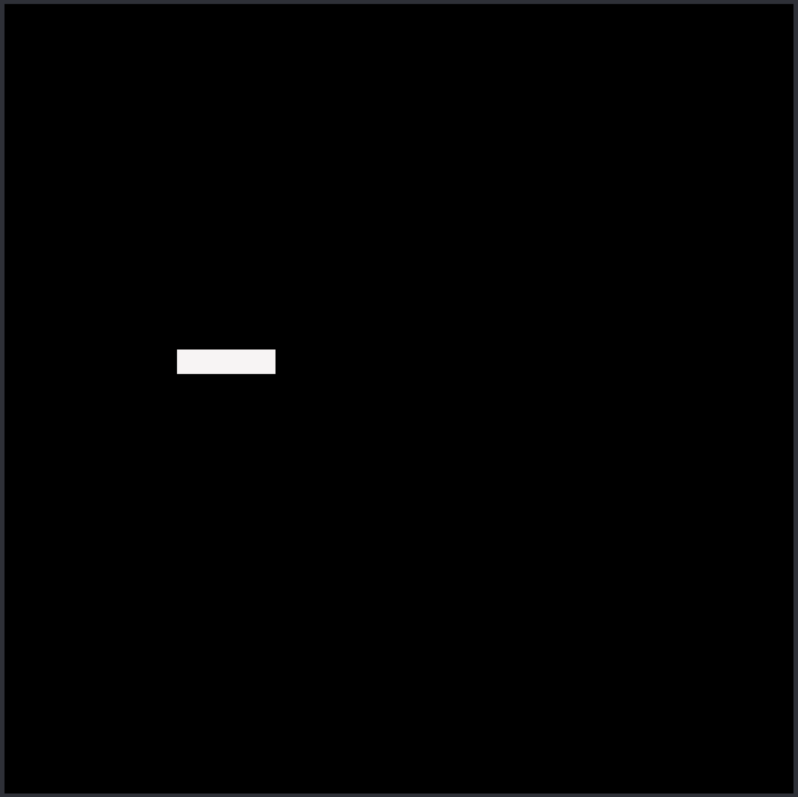
5. OUR PROJECT

6. CONCLUSION

The Game

Snake

1



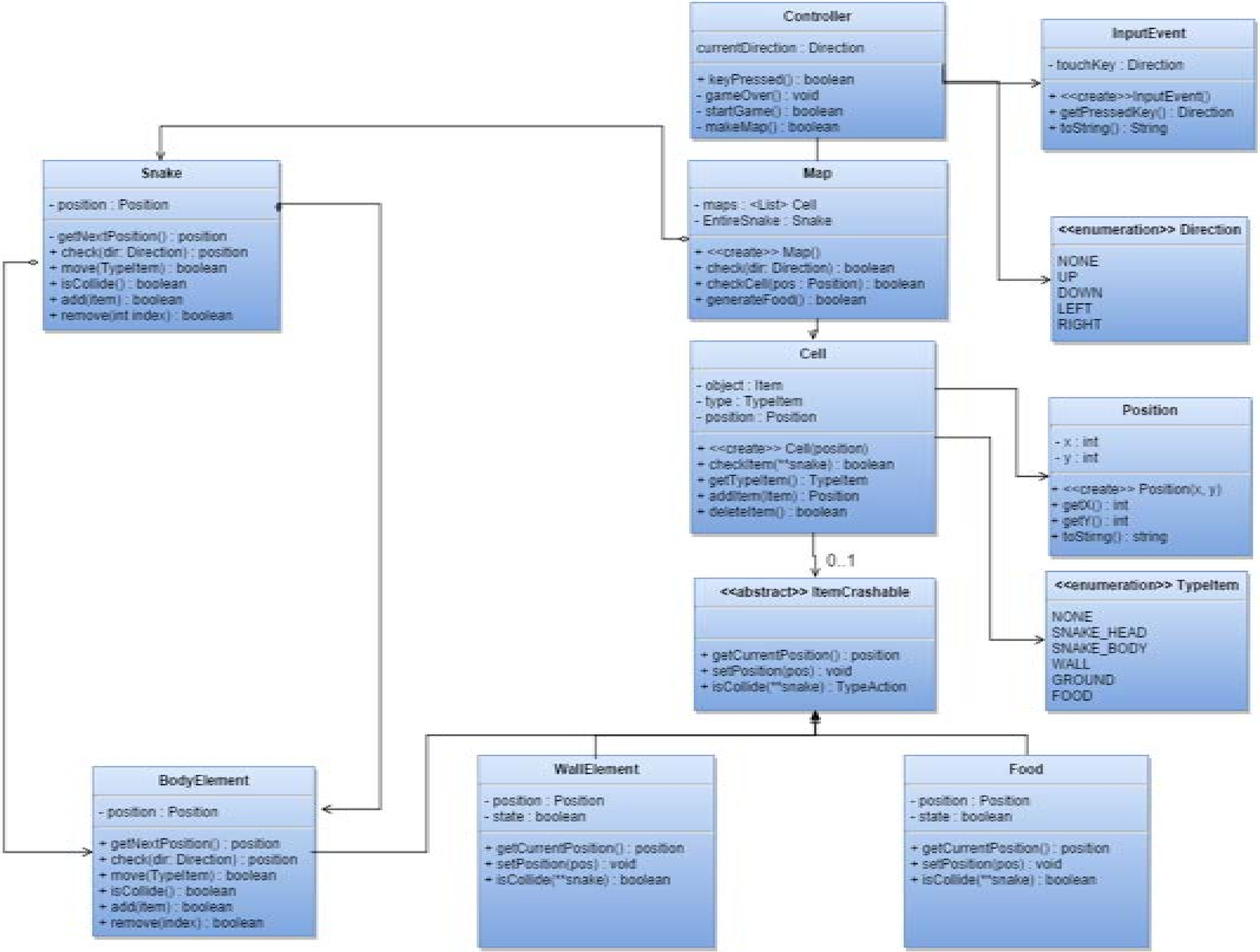
Snake History

The Snake is an old game create in 1979 by Peter Trefonas

The goal of this game is to eat the most fruit possible

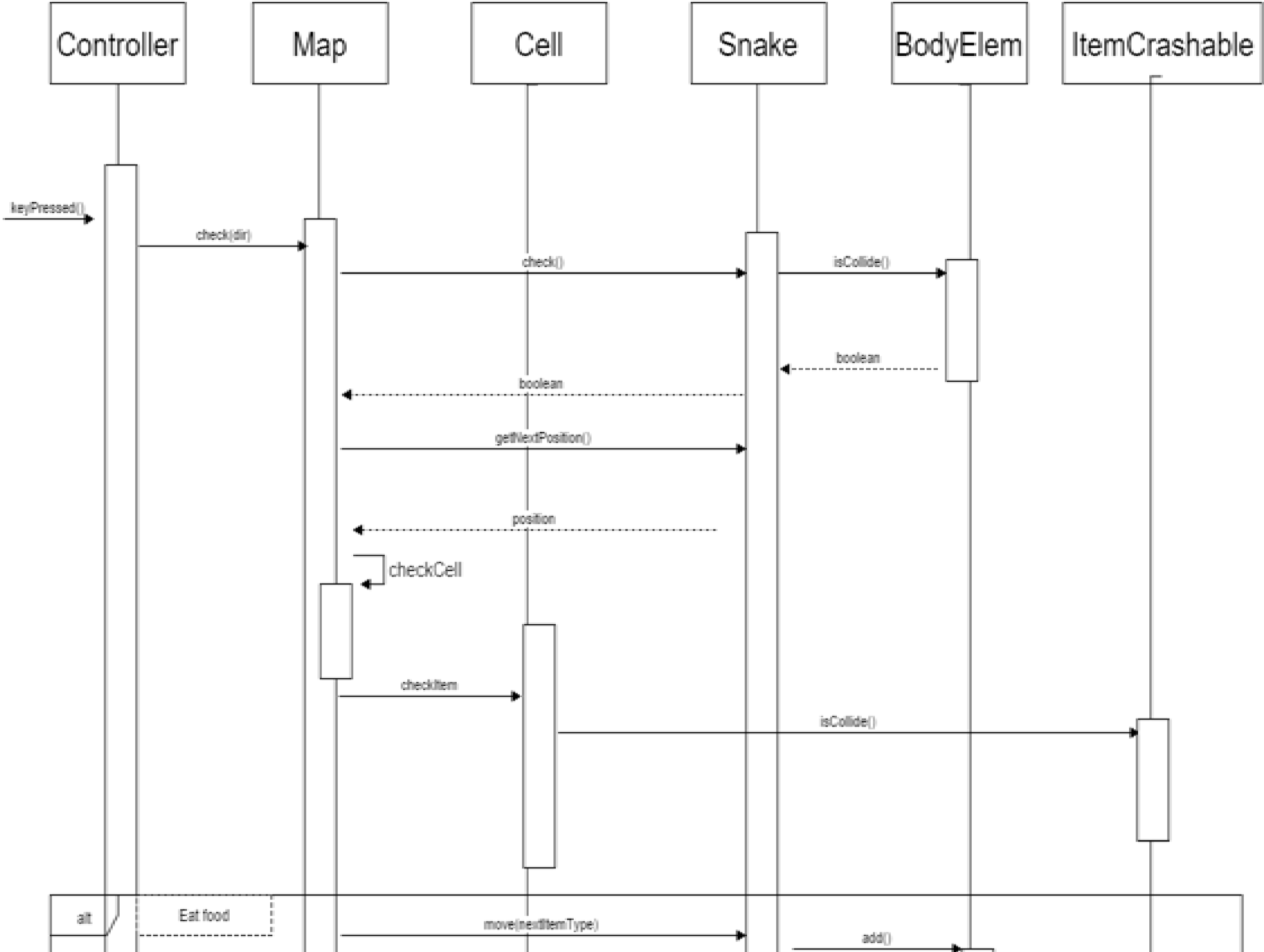
2

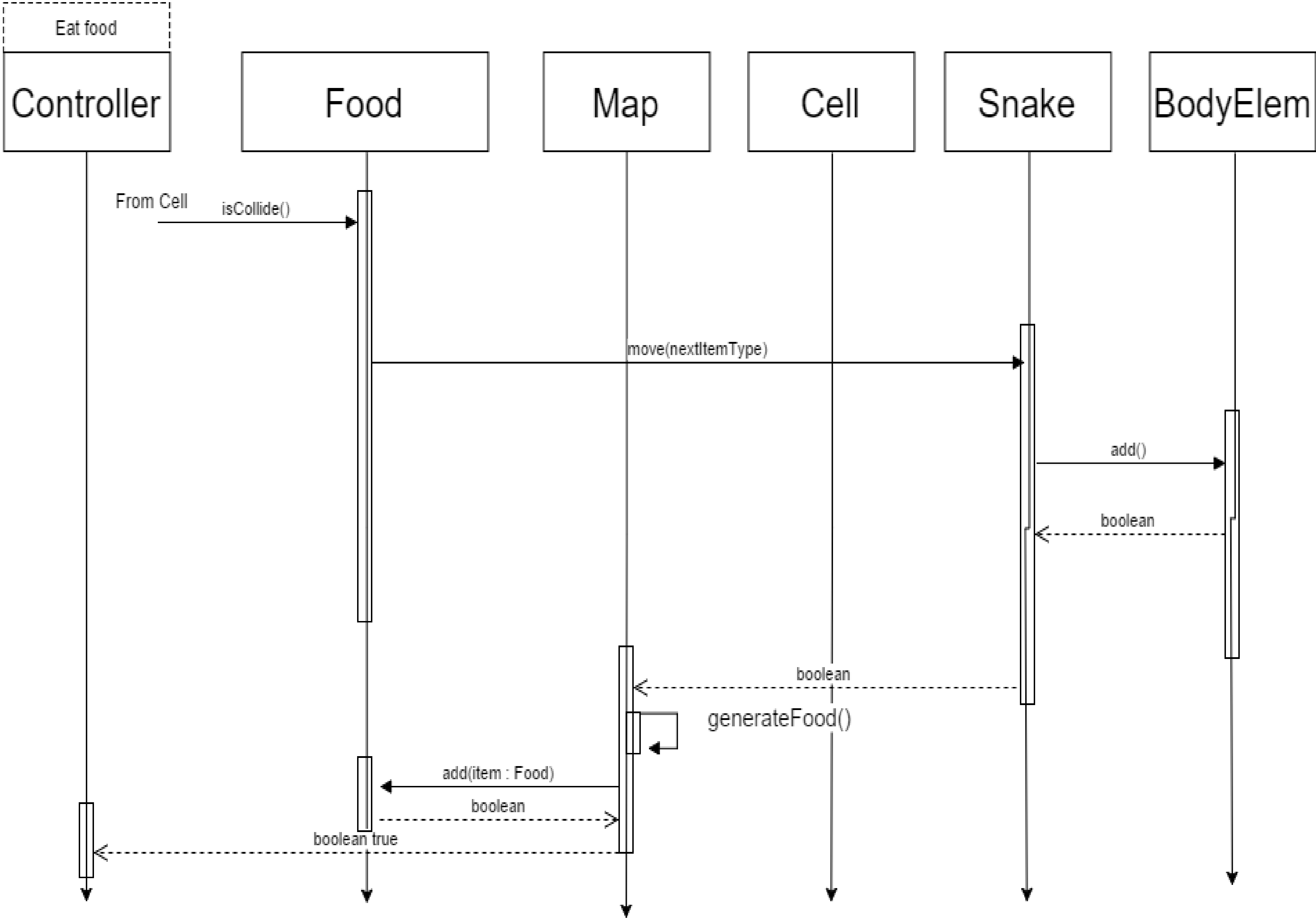
Class Diagram

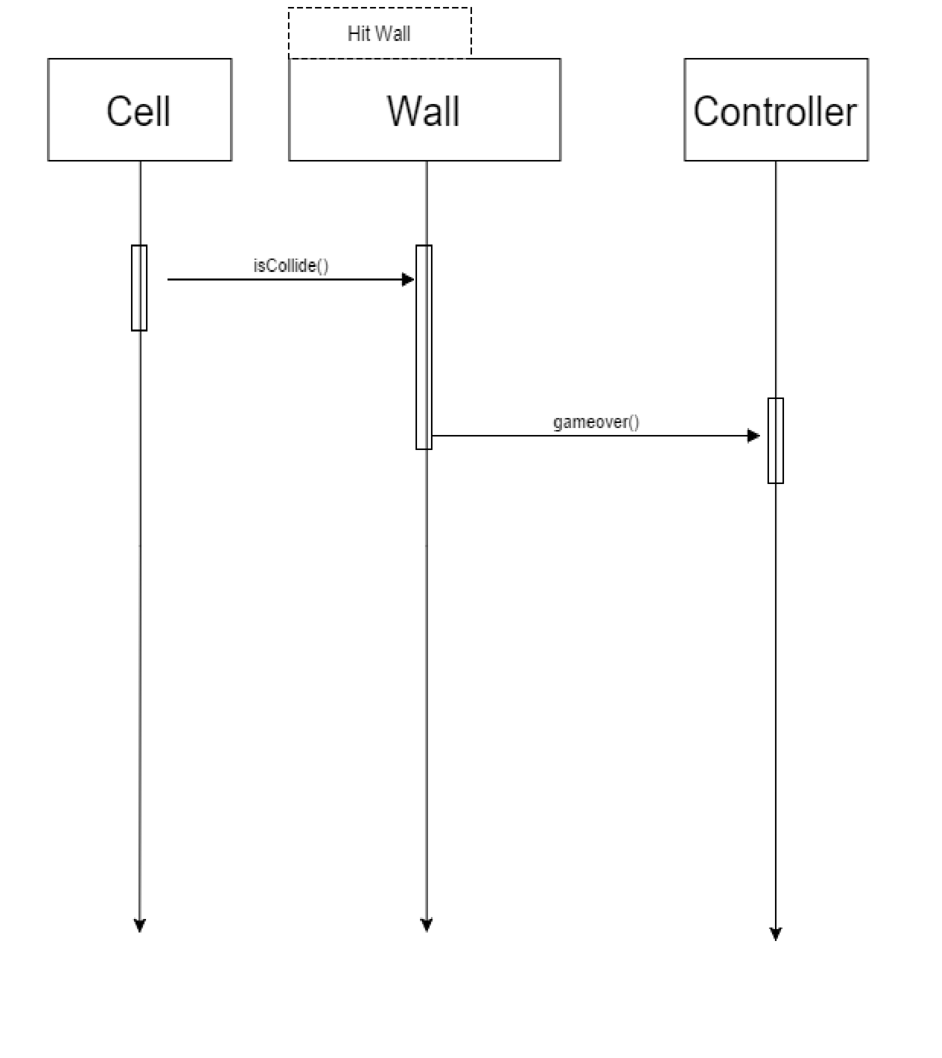

Controller
currentDirection : Direction
+ keyPressed() : boolean
- gameOver() : void
- startGame() : boolean
- makeMap() : boolean
InputEvent
- touchKey : Direction
+ <<create>>InputEvent()
+ getPressedKey() : Direction
+ toString() : String
Snake
- position : Position
- getNextPosition() : position
+ check(dir: Direction) : position
+ move(TypeItem) : boolean
+ isCollide() : boolean
+ add(item) : boolean
+ remove(int index) : boolean
Map
- maps : <List> Cell
- EntireSnake : Snake
+ <<create>> Map()
+ check(dir: Direction) : boolean
+ checkCell(pos : Position) : boolean
+ generateFood() : boolean
<<enumeration>> Direction
NONE
UP
DOWN
LEFT
RIGHT
Cell
- object : Item
- type : TypeItem
- position : Position
+ <<create>> Cell(position)
+ checkItem(**snake) : boolean
+ getTypeItem() : TypeItem
+ addItem(item) : Position
+ deleteItem() : boolean
Position
- x : int
- y : int
+ <<create>> Position(x, y)
+ getX() : int
+ getY() : int
+ toString() : string
0..1
<<abstract>> ItemCrashable
+ getCurrentPosition() : position
+ setPosition(pos) : void
+ isCollide(**snake) : TypeAction
<<enumeration>> TypeItem
NONE
SNAKE_HEAD
SNAKE_BODY
WALL
GROUND
FOOD
BodyElement
- position : Position
+ getNextPosition() : position
+ check(dir: Direction) : position
+ move(TypeItem) : boolean
+ isCollide() : boolean
+ add(item) : boolean
+ remove(index) : boolean
WallElement
- position : Position
- state : boolean
+ getCurrentPosition() : position
+ setPosition(pos) : void
+ isCollide(**snake) : boolean
Food
- position : Position
- state : boolean
+ getCurrentPosition() : position
+ setPosition(pos) : void
+ isCollide(**snake) : boolean

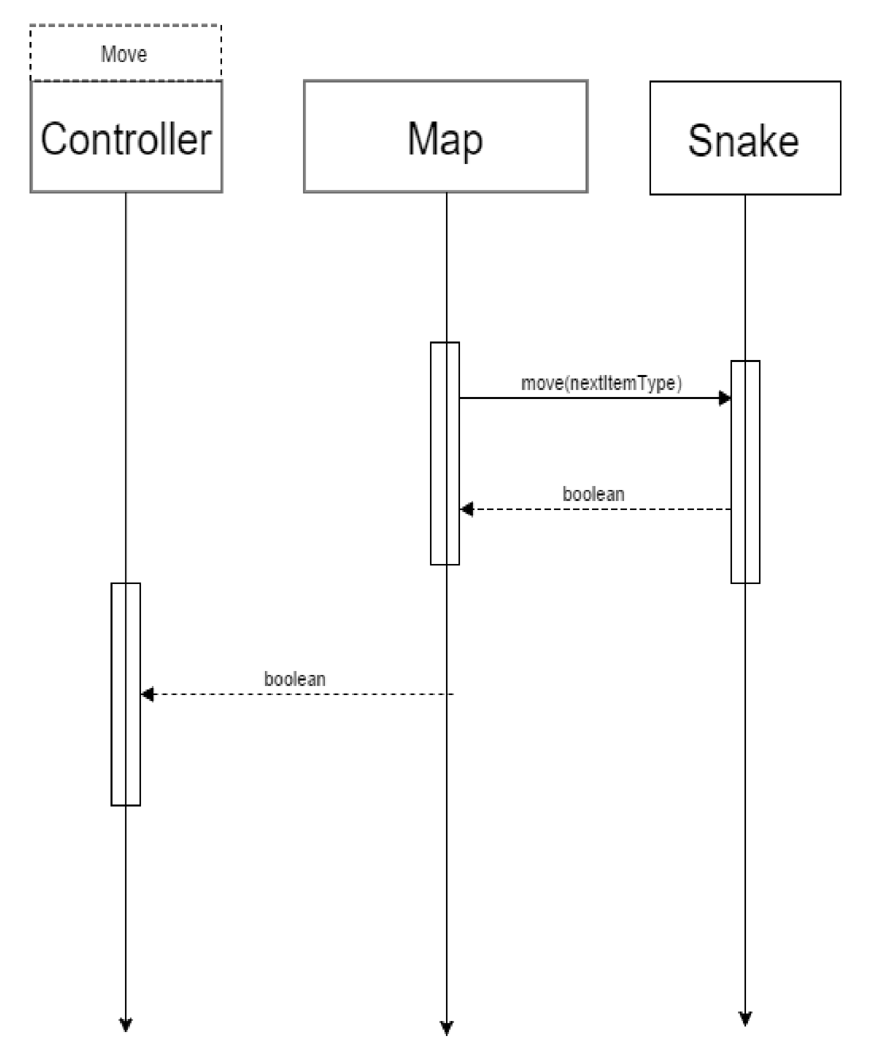
3

Sequence Diagrams


Controller
Map
Cell
Snake
BodyElem
ItemCrashable
keyPressed()
check(dir)
check()
isCollide()
boolean
boolean
getNextPosition()
position
checkCell
checkItem
isCollide()
alt
Eat food
move(nextItemType)
add()


Eat food
Controller
Food
Map
Cell
Snake
BodyElem
From Cell
isCollide()
move(nextItemType)
add()
boolean
boolean
generateFood()
add(item : Food)
boolean
boolean true


Hit Wall
Cell
Wall
Controller
isCollide()
gameover()

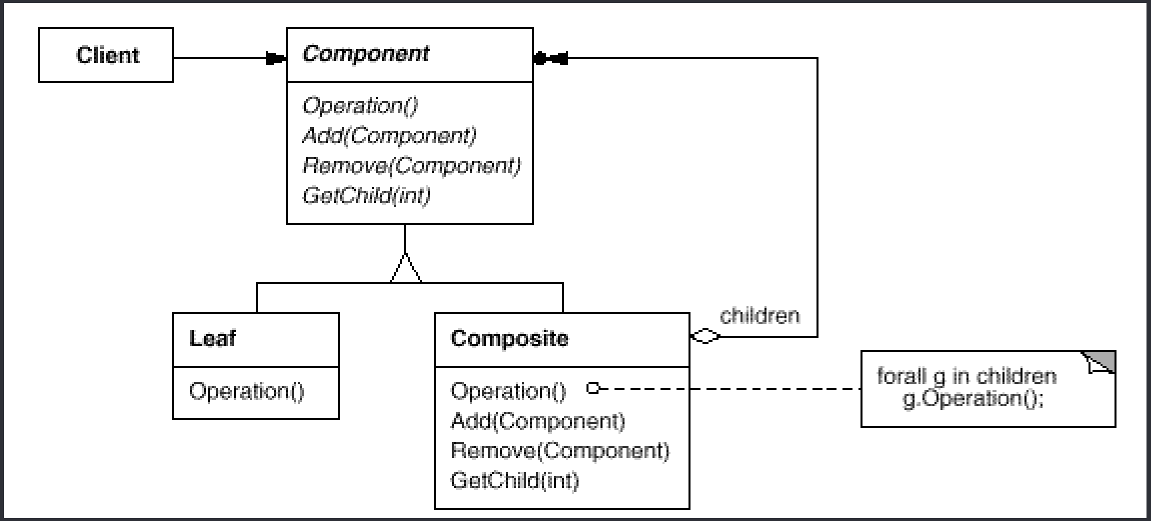

Move
Controller
Map
Snake
move(nextItemType)
boolean
boolean

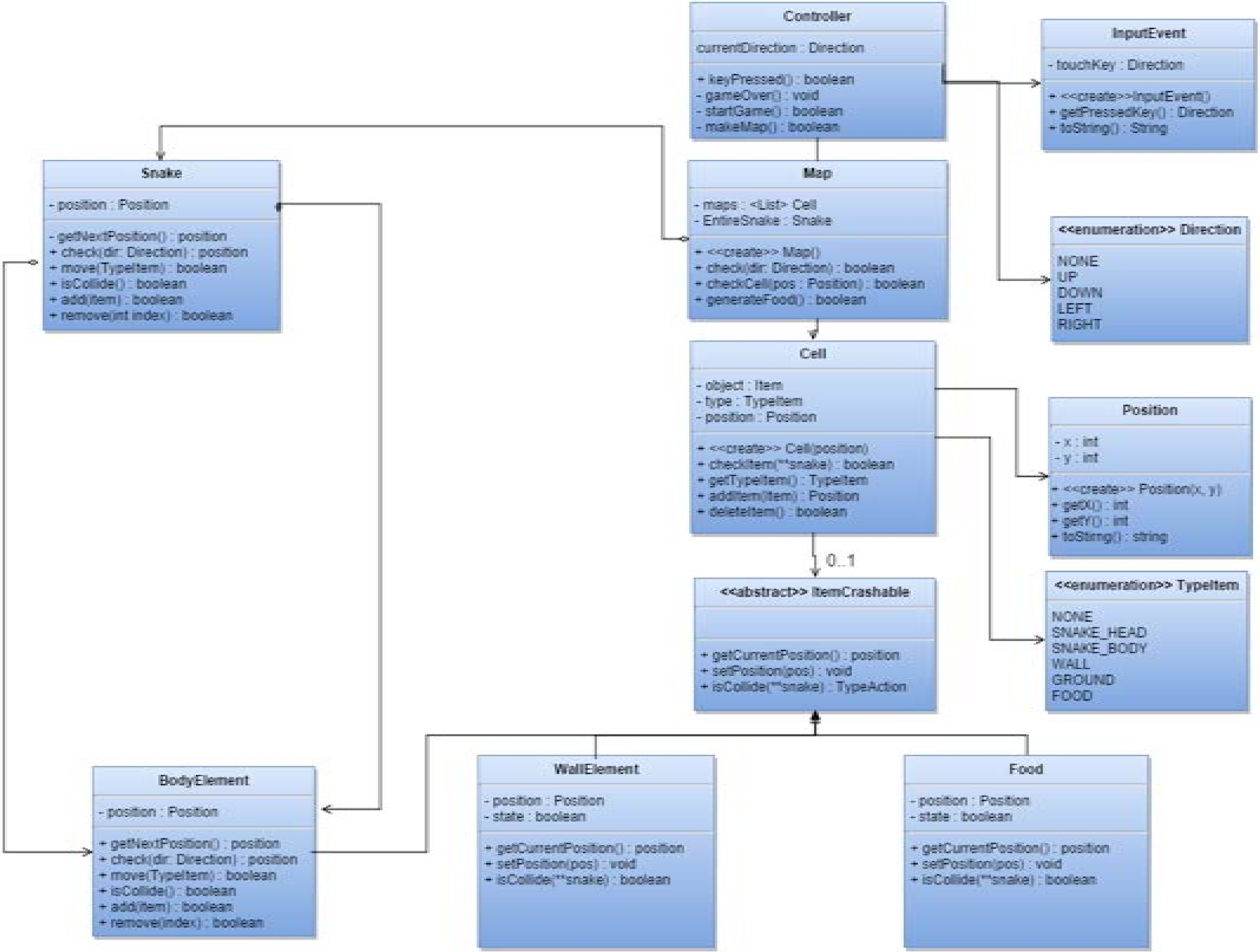
4

Design Pattern

◦ **Which Pattern ?**

- Composite Solution




Controller
currentDirection : Direction
+ keyPressed() : boolean
- gameOver() : void
- startGame() : boolean
- makeMap() : boolean
InputEvent
- touchKey : Direction
+ <<create>>InputEvent()
+ getPressedKey() : Direction
+ toString() : String
Snake
- position : Position
- getNextPosition() : position
+ check(dir: Direction) : position
+ move(TypeItem) : boolean
+ isCollide() : boolean
+ add(item) : boolean
+ remove(int index) : boolean
Map
- maps : <List> Cell
- EntireSnake : Snake
+ <<create>> Map()
+ check(dir: Direction) : boolean
+ checkCell(pos : Position) : boolean
+ generateFood() : boolean
<<enumeration>> Direction
NONE
UP
DOWN
LEFT
RIGHT
Cell
- object : Item
- type : TypeItem
- position : Position
+ <<create>> Cell(position)
+ checkItem(**snake) : boolean
+ getTypeItem() : TypeItem
+ addItem(item) : Position
+ deleteItem() : boolean
Position
- x : int
- y : int
+ <<create>> Position(x, y)
+ getX() : int
+ getY() : int
+ toString() : string
0..1
<<abstract>> ItemCrashable
+ getCurrentPosition() : position
+ setPosition(pos) : void
+ isCollide(**snake) : TypeAction
<<enumeration>> TypeItem
NONE
SNAKE_HEAD
SNAKE_BODY
WALL
GROUND
FOOD
BodyElement
- position : Position
+ getNextPosition() : position
+ check(dir: Direction) : position
+ move(TypeItem) : boolean
+ isCollide() : boolean
+ add(item) : boolean
+ remove(index) : boolean
WallElement
- position : Position
- state : boolean
+ getCurrentPosition() : position
+ setPosition(pos) : void
+ isCollide(**snake) : boolean
Food
- position : Position
- state : boolean
+ getCurrentPosition() : position
+ setPosition(pos) : void
+ isCollide(**snake) : boolean

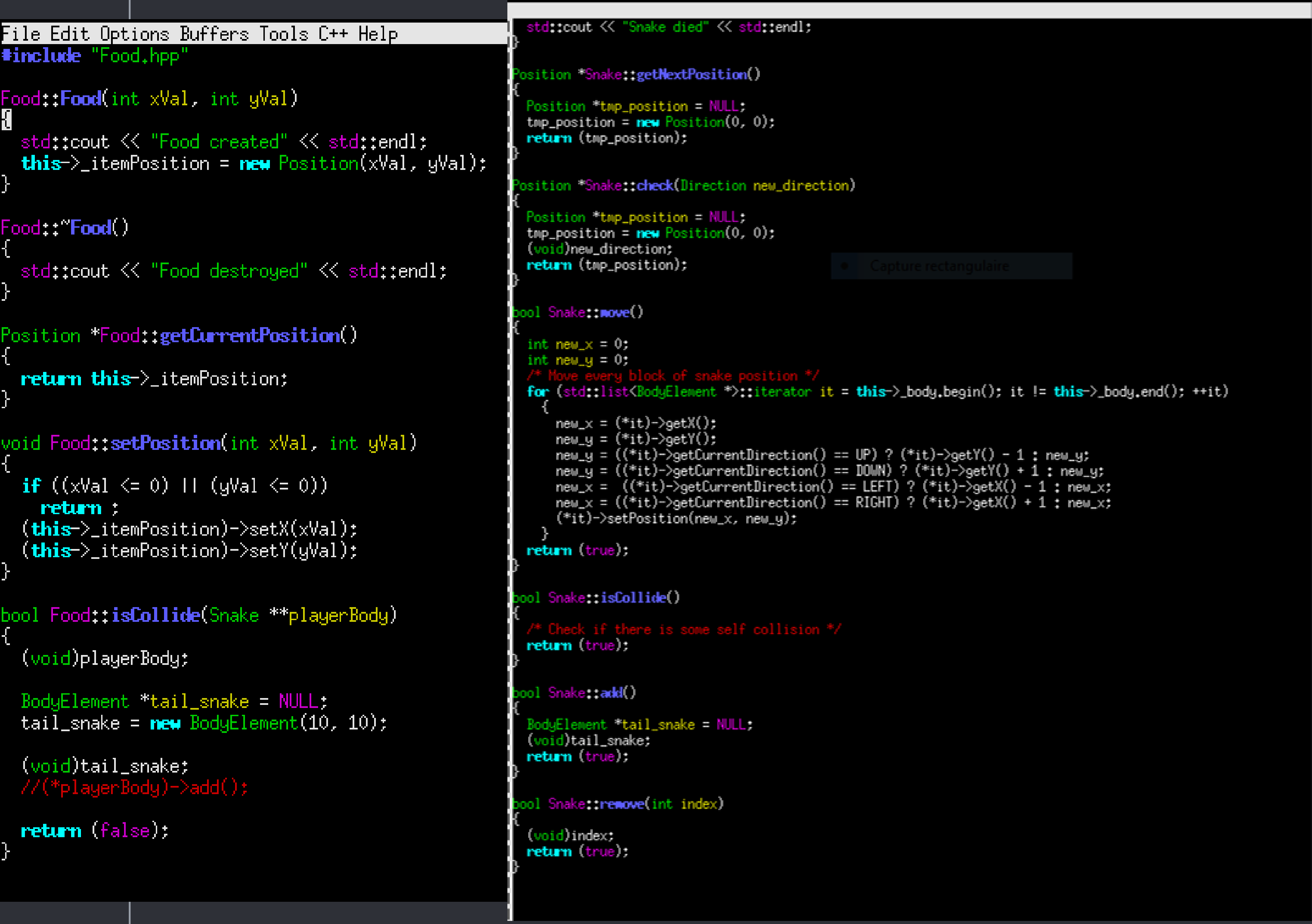
Composite Design Pattern

Why do we wanted to use it?

- It made the teamwork more flexible
- Have a maintainable code

5

Our Project



```cpp
#include "Food.hpp"

Food::Food(int xVal, int yVal)
{
  std::cout << "Food created" << std::endl;
  this->_itemPosition = new Position(xVal, yVal);
}

Food::~Food()
{
  std::cout << "Food destroyed" << std::endl;
}

Position *Food::getCurrentPosition()
{
  return this->_itemPosition;
}

void Food::setPosition(int xVal, int yVal)
{
  if ((xVal <= 0) || (yVal <= 0))
    return ;
  (this->_itemPosition)->setX(xVal);
  (this->_itemPosition)->setY(yVal);
}

bool Food::isCollide(Snake **playerBody)
{
  (void)playerBody;

  BodyElement *tail_snake = NULL;
  tail_snake = new BodyElement(10, 10);

  (void)tail_snake;
  //(*playerBody)->add();

  return (false);
}
```

```cpp
  std::cout << "Snake died" << std::endl;
}

Position *Snake::getNextPosition()
{
  Position *tmp_position = NULL;
  tmp_position = new Position(0, 0);
  return (tmp_position);
}

Position *Snake::check(Direction new_direction)
{
  Position *tmp_position = NULL;
  tmp_position = new Position(0, 0);
  (void)new_direction;
  return (tmp_position);
}

bool Snake::move()
{
  int new_x = 0;
  int new_y = 0;
  /* Move every block of snake position */
  for (std::list<BodyElement *>::iterator it = this->_body.begin(); it != this->_body.end(); ++it)
    {
      new_x = (*it)->getX();
      new_y = (*it)->getY();
      new_y = ((*it)->getCurrentDirection() == UP) ? (*it)->getY() - 1 : new_y;
      new_y = ((*it)->getCurrentDirection() == DOWN) ? (*it)->getY() + 1 : new_y;
      new_x =  ((*it)->getCurrentDirection() == LEFT) ? (*it)->getX() - 1 : new_x;
      new_x = ((*it)->getCurrentDirection() == RIGHT) ? (*it)->getX() + 1 : new_x;
      (*it)->setPosition(new_x, new_y);
    }
  return (true);
}

bool Snake::isCollide()
{
  /* Check if there is some self collision */
  return (true);
}

bool Snake::add()
{
  BodyElement *tail_snake = NULL;
  (void)tail_snake;
  return (true);
}

bool Snake::remove(int index)
{
  (void)index;
  return (true);
}
```

6 Conclusion

Conclusion

- Make the job easy within a team
- Provide an easy way to coding a program

CREDITS

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by [SlidesCarnival]
- Photographs by [Unsplash]

WeChat : axiang9428 (Arnaud)
xsylphe (Stéphane)
dasheels (Gabriel)