File - C:\untitled\app.cpp

```cpp
#include <chrono>
#include <thread>
#include <cstdlib>

#include "app.hpp"
#include "types.hpp"

using namespace std;


const int RENDER_SCALE = 5;
const int TIME_SCALE   = 50;


App::App(dim_t board_size, const int nb_snake)
{
    this->board_size = board_size;

    for (int i = 0; i < nb_snake; ++i)
    {
        this->snakes.push_back(
            new Snake(i, position_t(-1, -1), this)
        );
    }

    render = new Render(this->board_size, RENDER_SCALE);
}


```

File - C:\untitled\app.cpp

```
30 void App::initialize()
31 {
32     srand((unsigned)time(NULL));
33 
34     // Place les serpents aléatoirement et les pommes sur le plateau
35     for (int i = 0; i < this->snakes.size(); ++i)
36     {
37         Snake* snakeAt = nullptr;
38 
39         position_t snake_pos;
40 
41         do
42         {
43             snake_pos = randomBoardPosition();
44             snakeAt = getSnakeAtPosition(snake_pos.x, snake_pos.y);
45         } while (snakeAt != nullptr);
46 
47         this->snakes[i]->setPosition(snake_pos);
48 
49 
50         position_t apple_pos = randomBoardPosition();
51 
52         while (this->snakes[i]->isBody(apple_pos.x, apple_pos.y))
53         {
54             apple_pos = randomBoardPosition();
55         }
56 
57         this->snakes[i]->getPomme()->setPosition(apple_pos);
58     }
```

```
59 }
60
61
62 void App::run()
63 {
64     bool end = false;
65
66     while (!end)
67     {
68         // Events
69         SDL_Event e;
70         while (SDL_PollEvent(&e))
71         {
72             switch (e.type)
73             {
74                 case SDL_QUIT:
75                     end = true;
76                     break;
77                 default:
78                     break;
79             }
80         }
81
82         // Joue un tour de jeu
83         for (int i = 0; i < this->snakes.size(); ++i)
84         {
85             this->snakes[i]->update();
86         }
87
```

```
 88          //
 89          int nbSnakeAlive = 0;
 90          for (int i = 0; i < this->snakes.size(); ++i) {
 91              if (this->snakes[i]->getLength() > 0) {
 92                  ++nbSnakeAlive;
 93              }
 94          }
 95          if (nbSnakeAlive == 1) {
 96              end = true;
 97          }
 98
 99          // Dessine le plateau
100          this->render->draw_board();
101
102          for (int i = 0; i < this->snakes.size(); ++i)
103          {
104              this->snakes[i]->draw(render);
105          }
106
107          this->render->update();
108
109          // Attend x ms avant de jouer le prochain "tour"
110          this_thread::sleep_for(chrono::milliseconds(TIME_SCALE));
111      }
112
113
114
115 }
116
```

File - C:\untitled\app.cpp

```
117
118 dim_t App::getBoardSize()
119 {
120     return this->board_size;
121 }
122
123
124 App::~App()
125 {
126     for (int i = 0; i < this->snakes.size(); ++i)
127     {
128         delete this->snakes[i];
129     }
130
131     delete render;
132 }
133
134
135 position_t App::randomBoardPosition()
136 {
137     return position_t(
138         rand() % this->board_size.width,
139         rand() % this->board_size.height
140     );
141 }
142
143
144 Snake* App::getSnakeAtPosition(int x, int y)
145 {
```

```
146     Snake* r = nullptr;
147
148     for (int i = 0; i < this->snakes.size(); ++i)
149     {
150         position_t pos = this->snakes[i]->getPosition();
151
152         if (pos.x == x && pos.y == y)
153         {
154             r = this->snakes[i];
155             break;
156         }
157     }
158
159     return r;
160 }
161
162 int App::getNbrSnakes() {
163     return (int)this->snakes.size();
164 }
165
166 Snake* App::getSnake(int index) {
167     return this->snakes[index];
168 }
169
170
171
```

File - C:\untitled\app.hpp

```cpp
/*-----------------------------------------------------------------------
  Fichiers     : app.hpp et app.cpp
  Nom du labo  : TP8 - Snake
  Auteur(s)    : Ernst Laurent - Rodrigues Fraga Brian
  Date         : 20.01.2022
  But          : Nous souhaitons simuler des serpents allant chercher des pommes. Lors de leurs
                 déplacements, les serpents s'attaquent entre eux.
                 La partie se termine lorsque qu'un seul serpent est en jeu.

  Compilateur : gcc version 11.3.0
  IDE         : Clion 2022.3
  -----------------------------------------------------------------------*/

#ifndef app_hpp
#define app_hpp

#include <vector>

#include "types.hpp"
#include "render.hpp"
#include "snakes.hpp"

// Déclaration variable constante de taille serpent = 10
const int SNAKE_INIT_SIZE = 10;

class App
{
public:
```

File - C:\untitled\app.hpp

```
29      // ----- Constructeur ----- //
30      static App* instance;
31
32      App(dim_t board_size, const int nb_snake);
33
34      // ----- Destructeur ----- //
35
36      ~App();
37
38      // ----- Get et set ----- //
39
40      // Retourne la taille du plateau
41      dim_t getBoardSize();
42
43      // Retourne le serpent à la position x, y sinon retourne null
44      Snake* getSnakeAtPosition(int x, int y);
45
46      // Retourne le nombre de snakes
47      int getNbrSnakes();
48
49      // Retourne le pointeur du snake
50      Snake* getSnake(int index);
51
52      // ----- Méthodes ----- //
53
54      // Créer les serpents et le plateau
55      void initialize();
56
57      // Boucle de jeux
```

File - C:\untitled\app.hpp

```
58      void run();
59
60      // Retourne une position aléatoire sur le plateau
61      position_t randomBoardPosition();
62
63
64  private:
65      Render* render;
66
67      std::vector<Snake*> snakes;
68
69      dim_t board_size;
70  };
71
72  #endif
```

File - C:\untitled\main.cpp

```cpp
/*-----------------------------------------------------------------------
  Fichier     : main.cpp
  Nom du labo : TP8 - Snake
  Auteur(s)   : Ernst Laurent - Rodrigues Fraga Brian
  Date        : 20.01.2022
  But         : Nous souhaitons simuler des serpents allant chercher des pommes. Lors de
 leurs
                déplacements, les serpents s'attaquent entre eux.
                La partie se termine lorsque qu'un seul serpent est en jeu.

  Remarques   : Il est connu que dans les autres fichiers que le code affiche des warnings.
                Cela n'empêche pas le fonctionnement du jeu.

  Compilateur : gcc version 11.3.0
  IDE         : Clion 2022.3
  -----------------------------------------------------------------------*/

#include <iostream>
#include <iomanip>
#include <string>

#include "types.hpp"
#include "app.hpp"


using namespace std;

// ----- Déclaration des constantes ----- //
const int  MIN_LARGEUR = 50;
```

File - C:\untitled\main.cpp

```cpp
const int  MAX_LARGEUR = 1200;

const int  MIN_HAUTEUR = 50;
const int  MAX_HAUTEUR = 800;

const int  MIN_NBR_SNAKES = 2;
const int  MAX_NBR_SNAKES = 1000;

// ----- Fonctions ----- //
int askForValue(const string& message, const int min, const int max)
{
    int input;

    bool fail = true;
    while (fail)
    {
        cout << setw(12) << left << message << right << " [" << min << ".." << max << "] : "
;

        cin >> input;

        fail = cin.fail() || min > input || max < input;

        if (fail)
        {
            cin.clear();
            cout << "La valeur entrée est incorrect." << endl;
        }

```

File - C:\untitled\main.cpp

```
57          cin.ignore(numeric_limits<streamsize>::max(), '\n');
58      }
59
60      return input;
61  }
62
63  // ----- Début du programme ----- //
64  int main()
65  {
66      App* program;
67
68      cout << "Ce programme ..." << endl;
69
70      // Récupérer les paramètres de la partie
71      dim_t board_size;
72      board_size.width = askForValue("largeur", MIN_LARGEUR, MAX_LARGEUR);
73      board_size.height = askForValue("hauteur", MIN_HAUTEUR, MAX_HAUTEUR);
74
75      int nb_snake = askForValue("nbre serpent", MIN_NBR_SNAKES, MAX_NBR_SNAKES);
76
77
78      // Creation de la partie
79      program = new App(board_size, nb_snake);
80
81      // Initialization de la partie
82      program->initialize();
83
84      // Démarrage et déroulement de la partie
85      program->run();
```

```
86
87      // Fin de la partie et libération de la mémoire
88      delete program;
89
90      cout << "Fin du programme." << endl
91           << "Appuyer sur une touche pour quitter." << endl;
92
93      return EXIT_SUCCESS;
94  }
95
```

File - C:\untitled\types.hpp

```cpp
/*-----------------------------------------------------------------------------
  Fichier      : types.hpp
  Nom du labo : TP8 - Snake
  Auteur(s)    : Ernst Laurent - Rodrigues Fraga Brian
  Date         : 20.01.2022
  But          : Nous souhaitons simuler des serpents allant chercher des pommes. Lors de leurs
                 déplacements, les serpents s'attaquent entre eux.
                 La partie se termine lorsque qu'un seul serpent est en jeu.

  Compilateur : gcc version 11.3.0
  IDE          : Clion 2022.3
  ----------------------------------------------------------------------------*/

#ifndef types_hpp
#define types_hpp

// Permet de définir le type position_t avec un x et un y
struct position_t {
    int x;
    int y;

    position_t() { x = 0; y = 0; }
    position_t(int x, int y) : x(x), y(y) { }
};

// Permet de définir le type dim_t avec la largeur et la hauteur
struct dim_t {
    int width;
```

```
29      int height;
30
31      dim_t() { width = 0; height = 0; }
32      dim_t(int w, int h) : width(w), height(h) { }
33  };
34
35  // Permet de définir les mouvements du serpent
36  enum class MoveType { Left, Right, Up, Down };
37
38  #endif
```

File - C:\untitled\pommes.cpp

```cpp
#include "pommes.hpp"

// ----- Constructeur ----- //

Pomme::Pomme(position_t position, Snake* snake) {
    this->position = position;
    this->serpent = snake;
}


// ----- Get et set ----- //

position_t Pomme::getPosition() {
    return this->position;
}

void Pomme::setPosition(position_t position) {
    this->position = position;
}

Snake* Pomme::getSnake() {
    return this->serpent;
}

void Pomme::draw(Render* render) const
{
    SDL_Renderer* renderer = render->getRenderer();
    SDL_SetRenderDrawColor(renderer, 255, 0, 0, SDL_ALPHA_OPAQUE);
    SDL_RenderDrawPoint(renderer, position.x, position.y);
```

```
30 }
```

File - C:\untitled\pommes.hpp

```cpp
/*-----------------------------------------------------------------------
  Fichiers    : pommes.hpp et pommes.cpp
  Nom du labo : TP8 - Snake
  Auteur(s)   : Ernst Laurent - Rodrigues Fraga Brian
  Date        : 20.01.2022
  But         : Nous souhaitons simuler des serpents allant chercher des pommes. Lors de
 leurs
                déplacements, les serpents s'attaquent entre eux.
                La partie se termine lorsque qu'un seul serpent est en jeu.

  Compilateur : gcc version 11.3.0
  IDE         : Clion 2022.3
  -----------------------------------------------------------------------*/

#ifndef TP8_POMMES_HPP
#define TP8_POMMES_HPP

#include "types.hpp"
#include "render.hpp"

class Snake;

class Pomme {
public:
    // ----- Constructeur ----- //

    Pomme(position_t position, Snake* snake);

    // ----- Get et set ----- //
```

```
29
30      // Retourne la position de la pomme
31      position_t getPosition();
32
33      // Défini la position de la pomme
34      void setPosition(position_t position);
35
36      // Retourne le serpent
37      Snake* getSnake();
38
39      // ----- Méthodes ----- //
40
41      // Méthode exécuté après chaque frame (rendu graphique)
42      void draw(Render* render) const;
43
44  private:
45      Snake* serpent;
46
47      position_t position;
48  };
49
50  #endif //TP8_POMMES_HPP
51
```

File - C:\untitled\render.cpp

```cpp
#include "render.hpp"

using namespace std;

Render::Render(dim_t wsize, int scale)
{
    SDL_Init(SDL_INIT_VIDEO);
    SDL_CreateWindowAndRenderer(wsize.width * scale, wsize.height * scale, SDL_WINDOW_SHOWN
, &window, &renderer);

    SDL_SetWindowTitle(window, "Snake");
    SDL_RenderSetScale(renderer, scale, scale);
}


Render::~Render()
{
    if (renderer != nullptr)
    {
        SDL_DestroyRenderer(renderer);
    }

    if (window != nullptr)
    {
        SDL_DestroyWindow(window);
    }
}


```

File - C:\untitled\render.cpp

```
29 void Render::draw_board()
30 {
31     SDL_SetRenderDrawColor(renderer, 255, 255, 255, SDL_ALPHA_OPAQUE);
32     SDL_RenderClear(renderer);
33 }
34
35
36 void Render::update()
37 {
38     SDL_RenderPresent(renderer);
39 }
40
41
42 SDL_Renderer* Render::getRenderer()
43 {
44     return renderer;
45 }
46
47
48 SDL_Window* Render::getWindow()
49 {
50     return window;
51 }
```

File - C:\untitled\render.hpp

```cpp
/*-------------------------------------------------------------------------
  Fichiers    : render.hpp et render.cpp
  Nom du labo : TP8 - Snake
  Auteur(s)   : Ernst Laurent - Rodrigues Fraga Brian
  Date        : 20.01.2022
  But         : Nous souhaitons simuler des serpents allant chercher des pommes. Lors de leurs
                déplacements, les serpents s'attaquent entre eux.
                La partie se termine lorsque qu'un seul serpent est en jeu.

  Compilateur : gcc version 11.3.0
  IDE         : Clion 2022.3
  -------------------------------------------------------------------------*/

#ifndef render_hpp
#define render_hpp

#include <SDL.h>
#undef main

#include "types.hpp"

class Render
{
public:
    // ----- Constructeur ----- //

    Render(dim_t wsize, int scale);

```

```
29      // ----- Destructeur ----- //
30
31      ~Render();
32
33      // ----- Get et set ----- //
34
35      SDL_Renderer* getRenderer();
36
37      SDL_Window* getWindow();
38
39      // ----- Méthodes ----- //
40
41      void draw_board();
42
43      void update();
44
45  private:
46      SDL_Window* window;
47
48      SDL_Renderer* renderer;
49
50  };
51
52  #endif
```

File - C:\untitled\snakes.cpp

```cpp
#include <iostream>
#include <vector>

#include "snakes.hpp"
#include "app.hpp"

using namespace std;

// ----- Constructeur ----- //

Snake::Snake(unsigned int id, position_t position, App* app)
{
    this->id = id;
    this->app = app;

    for (int i = 0; i < 10; ++i)
    {
        this->positions.push_back(position);
    }

    pomme = new Pomme(position_t(0, 0), this);
};


// ----- Destructeur ----- //

Snake::~Snake()
{
    if (this->pomme != nullptr)
```

```
30      {
31          delete this->pomme;
32      }
33  }
34
35  // ----- Get et set ----- //
36
37  unsigned int Snake::getId() const {
38      return this->id;
39  }
40
41  int Snake::getLength() {
42      return (int) this->positions.size();
43  }
44
45  position_t Snake::getPosition() {
46      //position du serpent;
47      return this->positions[0];
48  }
49
50  void Snake::setPosition(position_t position) {
51      for (int i = 0; i < positions.size(); ++i)
52      {
53          positions[i] = position;
54      }
55  }
56
57  Pomme* Snake::getPomme() {
58      return pomme;
```

```
59 }
60
61
62 // ----- Méthodes ----- //
63
64 bool Snake::isHead(int x, int y)
65 {
66     return (positions.size() == 0) ? false : this->positions[0].x == x && this->positions[0
   ].y == y;
67 }
68
69 bool Snake::isBody(int x, int y)
70 {
71     bool r = false;
72
73     for (size_t i = 0; i < this->positions.size(); ++i) {
74         if (this->positions[i].x == x && this->positions[i].y == y) {
75             r = true;
76             break;
77         }
78     }
79
80     return r;
81 }
82
83 int Snake::split(int x, int y) {
84     int i = 0;
85     int split_index = -1;
86
```

```
 87     while (split_index == -1 && i < this->positions.size()) {
 88         if (this->positions[i].x == x && this->positions[i].y == y) {
 89             split_index = i;
 90         }
 91         ++i;
 92     }
 93
 94     int len = 0, splitted = 0;
 95     if (split_index != -1) {
 96         splitted = this->positions.size() - split_index;
 97         len = this->positions.size() - splitted;
 98         this->positions.resize(len);
 99     }
100     return splitted;
101 }
102
103 void Snake::move(MoveType direction)
104 {
105     if (positions.size() != 0)
106     {
107         position_t newPosition = positions[0];
108
109         switch (direction)
110         {
111             case MoveType::Up:
112                 newPosition.y -= 1;
113                 break;
114             case MoveType::Down:
115                 newPosition.y += 1;
```

```
116                 break;
117             case MoveType::Left:
118                 newPosition.x -= 1;
119                 break;
120             case MoveType::Right:
121                 newPosition.x += 1;
122                 break;
123             default:
124                 break;
125         }
126 
127         position_t oldPosition;
128         for (int i = 0; i < positions.size(); ++i)
129         {
130             oldPosition = positions[i];
131             positions[i] = newPosition;
132             newPosition = oldPosition;
133         }
134     }
135 }
136 
137 void Snake::kill(Snake* attacker) {
138     positions.resize(0);
139     cout << attacker->getId() << " killed " << this->getId() << endl;
140 }
141 
142 void Snake::addLength(int length) {
143     if (positions.size() > 0) {
144         for (int i = 0; i < length; ++i) {
```

```
145             positions.push_back(positions[positions.size() - 1]);
146         }
147     }
148 }
149
150 void Snake::update()
151 {
152     if (pomme != nullptr && positions.size() > 0)
153     {
154         position_t head = positions[0];
155         position_t posPomme = pomme->getPosition();
156
157         // On détermine la direction dans laquelle aller
158         position_t distance = position_t(posPomme.x - head.x, posPomme.y - head.y);
159
160         MoveType direction = MoveType::Right;
161
162         if (distance.x * distance.x >= distance.y * distance.y)
163         {
164             direction = (distance.x < 0) ? MoveType::Left : MoveType::Right;
165         }
166         else
167         {
168             direction = (distance.y < 0) ? MoveType::Up : MoveType::Down;
169         }
170
171         // On bouge le serpent dans la direction
172         this->move(direction);
173
```

```
174         // On vérifie si le serpent est sur la pomme
175         if (head.x == posPomme.x && head.y == posPomme.y)
176         {
177             // On ajoute 1 de longueur
178             positions.push_back(positions[positions.size() - 1]);
179 
180             // On change la position de la pomme
181             position_t posPomme = this->app->randomBoardPosition();
182 
183             while (head.x == posPomme.x && head.y == posPomme.y)
184             {
185                 posPomme = this->app->randomBoardPosition();
186             }
187 
188             pomme->setPosition(posPomme);
189         }
190 
191         // On vérifie si le serpent est sur un autre serpent
192         head = positions[0];
193         for (int i = 0; i < this->app->getNbrSnakes(); ++i) {
194             Snake* snake = this->app->getSnake(i);
195             if (this != snake) {
196                 if (snake->isBody(head.x, head.y)) {
197                     if (snake->isHead(head.x, head.y)) {
198                         if (positions.size() >= snake->getLength()) {
199                             int addLen = snake->getLength() * 0.60;
200                             addLength(addLen);
201                             snake->kill(this);
202                         } else {
```

```
203                             int addLen = this->getLength() * 0.60;
204                             snake->addLength(addLen);
205                             kill(snake);
206                         }
207                     } else {
208                         int splitted = snake->split(head.x, head.y);
209                         int addLen = splitted * 0.40;
210                         addLength(addLen);
211                     }
212                 }
213             }
214         }
215     }
216 }
217
218 void Snake::draw(Render* render)
219 {
220     SDL_Renderer* renderer = render->getRenderer();
221
222     SDL_SetRenderDrawColor(renderer, 0, 0, 0, SDL_ALPHA_OPAQUE);
223
224     for (int i = 0; i < positions.size(); ++i)
225     {
226         SDL_RenderDrawPoint(renderer, positions[i].x, positions[i].y);
227     }
228
229     if (pomme != nullptr && positions.size() > 0)
230     {
231         pomme->draw(render);
```

```
232     }
233 }
```

File - C:\untitled\snakes.hpp

```cpp
/*-------------------------------------------------------------------------
  Fichiers    : snakes.hpp et snakes.cpp
  Nom du labo : TP8 - Snake
  Auteur(s)   : Ernst Laurent - Rodrigues Fraga Brian
  Date        : 20.01.2022
  But         : Nous souhaitons simuler des serpents allant chercher des pommes. Lors de leurs
                déplacements, les serpents s'attaquent entre eux.
                La partie se termine lorsque qu'un seul serpent est en jeu.

  Compilateur : gcc version 11.3.0
  IDE         : Clion 2022.3
  -------------------------------------------------------------------------*/

#ifndef snakes_hpp
#define snakes_hpp

#include <vector>

#include "types.hpp"
#include "render.hpp"
#include "pommes.hpp"
class App;


class Snake {
public:
    // ----- Constructeur ----- //

```

```
29      Snake(unsigned int id, position_t position, App* app);
30
31      // ----- Destructeur ----- //
32
33      ~Snake();
34
35      // ----- Get et set ----- //
36
37      // Retourne l'ID du serpent
38      unsigned int getId() const;
39
40      // Retourne la longueur du serpent
41      int getLength();
42
43      // Retourne la position du serpent
44      position_t getPosition();
45
46      // Définie la position du serpent initial
47      void setPosition(position_t position);
48
49      // Retourne la pomme du serpent
50      Pomme* getPomme();
51
52      // ----- Méthodes ----- //
53
54      // Retourne true si le serpent occupe cette position
55      bool isBody(int x, int y);
56
57      // Méthode exécutée pour chaque frame
```

File - C:\untitled\snakes.hpp

```
58      void update();
59
60      // Méthode exécutée après chaque frame (rendu graphique)
61      void draw(Render* render);
62
63  private:
64      // ----- Variables ----- //
65
66      unsigned int id;
67
68      std::vector<position_t> positions;
69
70      Pomme* pomme;
71
72      App* app;
73
74      // ----- Méthodes ----- //
75
76      // Tue le serpent
77      void kill(Snake* attacker);
78
79      // Fait bouger le serpent dans une direction
80      void move(MoveType direction);
81
82      // Coupe le serpent à la position x, y
83      int split(int x, int y);
84
85      // Retourne true si la position x, y est la tête
86      bool isHead(int x, int y);
```

```
87
88      // Ajoute de la longueur au serpent
89      void addLength(int length);
90
91  };
92
93
94  #endif /* snakes_hpp */
95
```

File - C:\untitled\CMakeLists.txt

```
 1 cmake_minimum_required(VERSION 3.23)
 2 project(TP8)
 3
 4 set(CMAKE_CXX_STANDARD 20)
 5
 6 find_package(SDL2 REQUIRED)
 7 include_directories(${SDL2_INCLUDE_DIRS})
 8
 9 add_executable(TP8 main.cpp app.cpp render.cpp snakes.cpp pommes.cpp)
10
11 target_link_libraries(
12         TP8
13         ${SDL2_LIBRARIES}
14 )
```