CellWorld

Создано системой Doxygen 1.9.7

Глава 1

CellWorld

My first project on GitHub.

1.1 How to build project with vcpkg

Download required libraries with these commands:
vcpkg install glad[gl-api-33]:x64-windows
vcpkg install glfw3:x64-windows
vcpkg install imgui[core,glfw-binding,opengl3-binding]:x64-windows
vcpkg install eigen3:x64-windows
Download project, connect with vcpkg and done!

Глава 2

Алфавитный указатель групп

2.1 Группы

Полный список групп.

Глава 3

Иерархический список классов

3.1 Иерархия классов

Иерархия классов.

Глава 4

Алфавитный указатель классов

4.1 Классы

Классы с их кратким описанием.

Глава 5

Список файлов

5.1 Файлы

Полный список документированных файлов.

Глава 6

Группы

6.1 Коэффициенты

Набор коэффициентов, которые определяют правила мира.

Перечисления

- enum cellworld::Coefficients {
  cellworld::mass_into_energy , cellworld::mass_capacity , cellworld::starting_energy , cellworld::mass_cost ,
  cellworld::change_speed_module_cost , cellworld::braking_force , cellworld::mutation_strength ,
  cellworld::coefficients_count }

  Набор коэффициентов, которые определяют правила мира.

6.1.1 Подробное описание

Набор коэффициентов, которые определяют правила мира.

6.1.2 Перечисления

6.1.2.1 Coefficients

enum cellworld::Coefficients
Набор коэффициентов, которые определяют правила мира.

Элементы перечислений

mass_into_energy	Коэффициент преобразования массы в энергию. Сколько энергии требуется для создания единица массы. Используется при: • смерти сущности, энергия трупа+= масса∗mass_into_energy. • рождении сущности, энергия родителя-=масса ребёнка∗mass_into_energy.
mass_capacity	Коэффициент вместимости массы. Сколько энергии может хранить в себе единица массы. Макс энергия = mass_capacity ∗ масса существа.
starting_energy	Коэффициент начальной энергии. Определяет сколько энергии имеет существо при спавне(не рождении). Энергия = Макс энергия ∗ starting_energy.
mass_cost	Коэффициент стоимости массы. Определяет сколько энергии живое существо тратит за ход на поддержание единицы массы. Энергия-=масса∗mass_cost.

Элементы перечислений

change_speed_module_cost	Коэффициент стоимости изменения модуля скорости. Определяет сколько энергии существо тратит за ход на изменение модуля скорости. Энергия-=(изменение модуля скорости)^2*change_speed_module_cost.
braking_force	Коэффициент силы торможения. Определяет на сколько уменьшается модуль скорости в конце каждого хода. Модуль скорости-=braking_force.
mutation_strength	Коэффициент мутации. Определяет на сколько сильно будет отличатся геном ребёнка от родителя. • <=0 - копия родителя. • >=1 - геном не зависит от родителя совсем. Ген ребёнка=ген родителя*(1-mutation_strength) + случайный ген*mutation_strength.
coefficients_count	количество коэффициентов

6.2 Нейросеть

Все упоминания нейронных сетей

Определения типов

- using cellworld::NeuronNetwork = Eigen::Matrix< float, output_neurons_count,(4 *look_input_count+input_neu

  Хранит веса нейросети существа

Перечисления

- enum cellworld::InputNeurons { cellworld::pos_x , cellworld::pos_y , cellworld::speed_module , cellworld::energy , cellworld::bias , cellworld::input_neurons_count }

  Основные входные нейроны существа Содержит информацию о внутреннем состоянии существа
- enum cellworld::OutputNeurons { cellworld::change_speed_module , cellworld::vertical_or_horizontal , cellworld::decrease_or_increase , cellworld::reproduce , cellworld::output_neurons_count }

  Выходные нейроны существа
- enum cellworld::LookInput { cellworld::distance , cellworld::color_red , cellworld::color_green , cellworld::color_blue , cellworld::look_input_count }

  Дополнительные входные нейроны существа.

Функции

- std::uniform_real_distribution< float > cellworld::dis (-4.0f, 4.0f)

  Генератор случайных float в диапозоне [-4,4]. Определяет разброс значений у весов нейросети.

6.2.1 Подробное описание

Все упоминания нейронных сетей

6.2.2 Перечисления

6.2.2.1 InputNeurons

enum cellworld::InputNeurons
Основные входные нейроны существа Содержит информацию о внутреннем состоянии существа

Элементы перечислений

pos_x	Координата X.
pos_y	Координата Y.
speed_module	Модуль скорости
energy	Энергия
bias	Смещающий нейрон
input_neurons_count	Количество входных нейронов

6.2.2.2 LookInput

enum cellworld::LookInput
Дополнительные входные нейроны существа.
Содержит информацию об внешнем окружении существа.
Для каждого из 4 направлений(верх,низ,право,лево) существо имеет отдельный набор нейронов.
Поэтому общее количество входных нейронов равно $4 * look\_input\_count + input\_neurons\_count$.

Элементы перечислений

distance	расстояние от существа до ближайшего существа в одном из направлений
color_red	уровень красного цвета ближайшего существа
color_green	уровень зелёного цвета ближайшего существа
color_blue	уровень синего цвета ближайшего существа
look_input_count	Количество входных нейронов

6.2.2.3 OutputNeurons

enum cellworld::OutputNeurons
Выходные нейроны существа

Элементы перечислений

change_speed_module	Изменить модуль скорости
vertical_or_horizontal	По какой прямой перемещается существо, вертикальной или горизонтальной
decrease_or_increase	В какую сторону перемещается существо, по увеличению или уменьшению координаты
reproduce	Желание существа размножиться
output_neurons_count	Количество выходных нейронов

6.3 Существо

Классы

- class cellworld::Creature

Класс существа.

6.3.1 Подробное описание

ы

6.4 Поле

Все объекты связанные с полем.

Классы

- class cellworld::Field

  Класс Поля, хранит существ.

Обработка коллизий (ситуаций, когда 2 сущетсва претендуют на 1 позицию)

- void cellworld::conjoin (Creature *&champion, Creature *&candidate)
- void cellworld::conjoin (Creature &champion, Creature &candidate)

6.4.1 Подробное описание

Все объекты связанные с полем.

6.4.2 Функции

6.4.2.1 conjoin() [1/2]

void cellworld::conjoin (
Creature & master,
Creature & candidate)

Аргументы

master	- существо, занявшее эту позицию первым
candidate	- существо, занявшее эту позицию вторым

6.4.2.2 conjoin() [2/2]

void cellworld::conjoin (
Creature *& master,
Creature *& candidate)

Аргументы

master	- существо, занявшее эту позицию первым
candidate	- существо, занявшее эту позицию вторым

6.5 сохранений

Классы

- class cellworld::FileSystem

  Класс хранения файлов.

6.5.1 Подробное описание

Глава 7

Классы

7.1 Класс cellworld::Creature

Класс существа.
#include <creature.h>

Открытые члены

- Creature (const Creature &)=default
- Creature (Creature &&)=default
- Creature & operator= (const Creature &)=default
- Creature & operator= (Creature &&)=default
- float Leftover ()

  Возвращает избыток энергии существа. Энергия уменьшается до лимита энергии.
- bool wantToReproduce () const

  проверка на желание размножиться
- void look (Creature &found, int direction)

  Получение информации о соседнем существе.
- void getInfo ()

  Получение информации о себе.
- void reverseInput ()

  Нормализация входных данных.
- void think ()

  Работа нейросети.
- void act ()

  Работа с выходными нейронами.
- void eat (Creature &)

  Существо забирает всю энергию другого сущетсва.
- void addEnergy (const float &energy)

  Существо получает энергию в размере energy.
- void buildIO ()

  Инициализирует недостающие элементы существа при чтении из файла.

Специальные конструкторы существа

По умолчанию модуль скорости равен 0, существо направлено вверх.

- Creature (const Genome &gen, const Position &pos)

  Конструируется живое существо
- Creature ()

  Конструируется несущетвующее существо
- Creature (float energy, const Position &pos)

Конструируется неживое существо

Изменение состояния существа

Используются для избежания лишней инициализации

- void makeAlive (Creature &ancestor, const Position &pos)
  Существо становится живым
- void makeAlive (const Position &pos)
  Существо становится живым
- void die ()
  Существо становится мёртвым.
- void stopExisting ()
  Существо перестаёт существовать.

Геттеры

- const int & getState () const
- const int & getDirection () const
- const int & getSpeed () const
- int & getX ()
  Возвращат ссылку на изменяемое значение, равное, но не являющееся координатой X.
- int & getY ()
  Возвращат ссылку на изменяемое значение, равное, но не являющееся координатой Y.
- const float & getEnergy () const
- const float & getEnergyLimit () const
- const int & getMass () const
- const unsigned int & getColor () const
- unsigned int getBlue () const
- unsigned int getGreen () const
- unsigned int getRed () const
- const Genome & getGenome ()

Открытые статические члены

- static Genome generateGenome ()
  Создать случайным геном
- static void generateGenome (Genome &)
  Сделать данный геном случайным
- static void mixGen (float &gen1, const float &gen2)
- static unsigned int mixGen (const unsigned int &gen1, const unsigned int &gen2)
- static Genome createGenome (const Genome &ancestor)
- static unsigned int energyColor (int energy)

Открытые атрибуты

Позиция существа

Находится в public, чтобы синхронизировать позицию, как внутреннее состояние сущности, и позицию, как индекс массива.

- int pos_x_
- int pos_y_

Статические открытые данные

- static std::array< float, coefficients_count > coeff_ { 0 }
  Массив, который хранит значения всех коэффициентов.
- static bool is_breedable = 1
  Определяет включена ли возможность размножения, если 0 - выключена, иначе - включена.
- static unsigned int base_color_ = 0xAfAfAfff
  Цвет несуществующего существа.

7.1.1 Подробное описание

Класс существа.

7.1.2 Конструктор(ы)

7.1.2.1 Creature() [1/2]

cellworld::Creature::Creature (
const Genome & gen,
const Position & pos)

Конструируется живое существо

Аргументы

gen,pos	- геном и позиция существа соответсвенно. начальная энергия зависит веса(см. Коэффициенты)

7.1.2.2 Creature() [2/2]

cellworld::Creature::Creature (
float energy,
const Position & pos)

Конструируется неживое существо

Аргументы

energy,pos	- энергия и позиция существа соответсвенно.

7.1.3 Методы

7.1.3.1 act()

void cellworld::Creature::act ( )

Работа с выходными нейронами.
В зависимости от выходных нейронов меняет модуль скорости и направление.(если существо живое)
После чего обновляется положение существа. В конце функции модуль скорости уменьшается на коэффициент торможения.

7.1.3.2 createGenome()

Genome cellworld::Creature::createGenome (
const Genome & ancestor) [static]

/brief Создать геном, основанный на геноме предка.

7.1.3.3 die()

void cellworld::Creature::die ( )

Существо становится мёртвым.
Вес преобразовывается в энергию(см. Коэффициенты).

7.1.3.4 energyColor()

unsigned int cellworld::Creature::energyColor (
int energy) [static]

/brief Цвет энергии. Цвет мёртового существа зависит от его текущей энергии.

7.1.3.5 getInfo()

void cellworld::Creature::getInfo ( )
Получение информации о себе.
Для большей информации см. InputNeuron

7.1.3.6 getX()

int & cellworld::Creature::getX ( )
Возвращат ссылку на изменяемое значение, равное, но не являющееся координатой X.

То есть, при изменении значения данной ссылки координаты не меняются. Сделано так криво, чтобы спокойно работать в многопотоке.

7.1.3.7 getY()

int & cellworld::Creature::getY ( )
Возвращат ссылку на изменяемое значение, равное, но не являющееся координатой Y.

При изменении данной значение данной ссылки координаты не меняются. Сделано так криво, чтобы спокойно работать в многопотоке.

7.1.3.8 look()

void cellworld::Creature::look (
Creature & found,
int direction)
Получение информации о соседнем существе.

Аргументы

found	- обнаруженное существо.
direction	- направление, в котором обнаружено данное существа. Для большей информации см. LookInput

7.1.3.9 makeAlive() [1/2]

void cellworld::Creature::makeAlive (
const Position & pos)
Существо становится живым

Аргументы

pos	- позиция существа. Геном рандомный.

7.1.3.10 makeAlive() [2/2]

void cellworld::Creature::makeAlive (
Creature & ancestor,
const Position & pos)
Существо становится живым

Аргументы

ancestor	- предок существа.
pos	- позиция существа. Геном основывается на предке.

7.1.3.11 mixGen() [1/2]

unsigned int cellworld::Creature::mixGen (
const unsigned int & gen1,
const unsigned int & gen2) [static]

/brief Смешивание генов

Аргументы

gen1	- главный ген
gen2	- случайный ген

7.1.3.12 mixGen() [2/2]

void cellworld::Creature::mixGen (
float & gen1,
const float & gen2) [static]

/brief Смешивание генов

Аргументы

gen1	- главный ген
gen2	- случайный ген

7.1.3.13 reverseInput()

void cellworld::Creature::reverseInput ( )

Нормализация входных данных.
Переводит в диапозон [0;1] путём быстрого, но неточного деления.

7.1.3.14 stopExisting()

void cellworld::Creature::stopExisting ( ) [inline]

Существо перестаёт существовать.
Вес преобразовывается в энергию(см. Коэффициенты).

7.1.3.15 think()

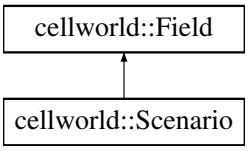
void cellworld::Creature::think ( )

Работа нейросети.
Обрабатывает входные нейроны с помощью весов, расположенных в геноме существа, и выводит результат в выходные нейроны.
Объявления и описания членов классов находятся в файлах:

- objects/creature/include/creature/creature.h
- objects/creature/creature.cpp

7.2 Класс cellworld::Field

Класс Поля, хранит существ.
#include <creature.h>
Граф наследования:cellworld::Field:



Открытые члены

- Field (int size_x=0, int size_y=0)

  Конструктор класса с заданным размером.
- void updatePositions ()

  Обновляет позиции существ.
- void updateStates ()

  Обновляет состояние существ. Если у живого существа <0 энергии - оно умирает. Если существо хочет размножится, то одно из сосодних существ становится живым, с геном, наследуемым от предка. Работает в многопоточном режиме, без инициализаций.
- Position findClosePosition (Creature *ancestor)

  Соседняя позиция относительно существа, желающего размножиться, которая становится живой. Если позиция не найдена, возвращает bad_position.
- Creature & findCreature (Creature *finder, int direction)

  Поиск существа из определённого сущетсва в определённом направлении.
- void clear ()

  Очистка поля

Доступ к существу по 1 или 2 индексам(позиции) поля

При доступе по 2 индексам происходит проверка на валидность позиции, если позиция не валидна возвращается bad_creature.

- const Creature & getCreature (const Position &pos) const
- Creature & getCreature (const Position &pos)
- const Creature & getCreature (const int &index) const
- Creature & getCreature (const int &index)
- const Creature & operator[] (const int &index) const
- Creature & operator[] (const int &index)

Геттеры

- unsigned int getColor (const Position &pos) const
- unsigned int getColor (const int &index) const
- int sizeX () const
- int sizeY () const
- int size () const

Проверка координаты на валидность

- bool validX (const int &x) const
- bool validY (const int &y) const

Интерфейс визуализации

- void createTexture ()
- void * getTexture ()

  Получить текстуру для работы с ImGui.
- const GLuint & getGLTexture ()

  Получить текстуру для работы с OpenGL.
- void updateTexture ()

  Синхронизировать текстуру с текущим состоянием поля.
- void unbindTexture ()

  Закончить синхронизацию с полем.

Статические открытые данные

- static Creature bad_creature =Creature()

Защищенные данные

- std::vector< Creature * > zoo_ptr_

  Указатели на текущее состояние поля
- std::vector< Creature * > empty_zoo_ptr_

  Указатели на запасное состояние поля
- std::vector< Creature > storage_

  Хранилище существ
- std::vector< unsigned int > colors_

  Хранит цвета каждого существа, используется при создании текстуры.
- GLuint texture_

  Указатель на текстуру

7.2.1 Подробное описание

Класс Поля, хранит существ.
Задачи поля:

- Хранить конкретное состояние
- Обновлять состояние
- Контролировать поведение, состояние, взаимодействие существ
- Визуализировать симуляцию

7.2.2 Конструктор(ы)

7.2.2.1 Field()

```
cellworld::Field::Field (
            int size_x = 0,
            int size_y = 0 )
```

Конструктор класса с заданным размером.

Аргументы

size_x	- ширина
size_y	- высота

7.2.3 Методы

7.2.3.1 createTexture()

```
void cellworld::Field::createTexture ( )
```

Создать текстуру

7.2.3.2 findCreature()

```
Creature & cellworld::Field::findCreature (
            Creature * finder,
            int direction )
```

Поиск существа из определённого сущетсва в определённом направлении.

Аргументы

finder	- существо, из которого ведётся поиск
direction	- направление, в котором ведётся поиск.

7.2.3.3 updatePositions()

void cellworld::Field::updatePositions ( )
Обновляет позиции существ.
Ввод данных в нейросеть. Происходит вся работа нейросетей. Забирает энергии у существ за передвижение. Оставляет на предыдущей позиции остатки энергии. Обрабатывет коллизию существ. Работает в многопоточном режиме, без инициализаций.

7.2.3.4 updateStates()

void cellworld::Field::updateStates ( )
Обновляет состояние существ. Если у живого существа <0 энергии - оно умирает. Если существо хочет размножится, то одно из сосодних существ становится живым, с геном, наследуемым от предка. Работает в многопоточном режиме, без инициализаций.
Объявления и описания членов классов находятся в файлах:

- objects/creature/include/creature/creature.h
- objects/creature/creature.cpp

7.3 Класс cellworld::FileSystem

Класс хранения файлов.
#include <save_system.h>

Открытые члены

- FileSystem (std::string name)

  Создаёт экземпляр, с файлом, хранящим имена, с именем name+".txt". "file_names.txt".
- FileSystem ()

  Создаёт экземпляр, с файлом, хранящим имена, с именем "file_names.txt".
- void addFileName (const char *file_name)

  Добавить в экземпляр, файл с информацией с именем file_name.
- void removeFileName (int index)

  Удалить имя файла из массива имён с данным индексом, и удалить сам файл из операционной системы.
- void saveFileNames ()

  Сохранить массив имён в текстовый файл.
- void loadFileNames ()

  Загрузить массив имён из текстового файла.
- void checkFileNames ()

  Проверить валидность каждого из имён файлов.
- bool findFileName (const char *file_name)

  Проверить наличие данного имени в массиве.
- std::string getValidFileName (std::string file_name)

  Получить неиспользуемое имя, схожее с данным.
- const std::vector< std::string > & getFileNames ()

  Доступ к массиву имён.

7.3.1 Подробное описание

Класс хранения файлов.
Состоит из 2 частей:

- Файлы, содержащие информацию.
- Текстовый файл, хранящий имена файлов с информацией.

Объявления и описания членов классов находятся в файлах:

- objects/save_system/include/save_system/save_system.h
- objects/save_system/save_system.cpp

7.4 Структура cellworld::Genome

Хранит геном существа.
#include <creature.h>

Открытые члены

- Genome (unsigned int in_color)

Открытые атрибуты

- unsigned int color
- unsigned int mass
- NeuronNetwork neuron_network

7.4.1 Подробное описание

Хранит геном существа.
В геном входят:

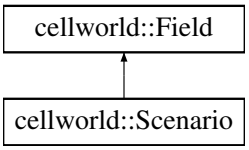
- цвет
- масса
- веса нейросети существа

Объявления и описания членов структуры находятся в файле:

- objects/creature/include/creature/creature.h

7.5 Класс cellworld::Scenario

Граф наследования:cellworld::Scenario:



Открытые члены

- Scenario (int size_x, int size_y)
- void makeNew ()
- void makeOneStep ()
- void spawnCreatures (int amount)
- void makeRewards (Position, Position, float strength)

- void giveRewards ()
- void CancelRewardsChange ()
- void resetRewards ()
- void rewardsEditor (ImVec2 window_pos, ImVec2 window_size, float strength, ImTextureID texture)
- void updateRewardsTexture ()
- void newCycle ()
- int getInitialPopulation ()
- long long getIteration ()

Открытые члены унаследованные от cellworld::Field

- Field (int size_x=0, int size_y=0)

  Конструктор класса с заданным размером.
- void updatePositions ()

  Обновляет позиции существ.
- void updateStates ()

  Обновляет состояние существ. Если у живого существа <0 энергии - оно умирает. Если существо хочет размножится, то одно из сосодних существ становится живым, с геном, наследуемым от предка. Работает в многопоточном режиме, без инициализаций.
- Position findClosePosition (Creature *ancestor)

  Соседняя позиция относительно существа, желающего размножиться, которая становится живой. Если позиция не найдена, возвращает bad_position.
- Creature & findCreature (Creature *finder, int direction)

  Поиск существа из определённого сущетсва в определённом направлении.
- void clear ()

  Очистка поля

- const Creature & getCreature (const Position &pos) const
- Creature & getCreature (const Position &pos)
- const Creature & getCreature (const int &index) const
- Creature & getCreature (const int &index)
- const Creature & operator[] (const int &index) const
- Creature & operator[] (const int &index)

- unsigned int getColor (const Position &pos) const
- unsigned int getColor (const int &index) const
- int sizeX () const
- int sizeY () const
- int size () const

- bool validX (const int &x) const
- bool validY (const int &y) const

- void createTexture ()
- void * getTexture ()

  Получить текстуру для работы с ImGui.
- const GLuint & getGLTexture ()

  Получить текстуру для работы с OpenGL.
- void updateTexture ()

  Синхронизировать текстуру с текущим состоянием поля.
- void unbindTexture ()

  Закончить синхронизацию с полем.

Открытые атрибуты

- int cycle_len_

Друзья

- void saveWorld (const char *path, Scenario *current_field, unsigned int seed)
- void loadWorld (const char *path, Scenario *current_field, unsigned int &seed)

Дополнительные унаследованные члены

Статические открытые данные унаследованные от cellworld::Field

- static Creature bad_creature =Creature()

Защищенные данные унаследованные от cellworld::Field

- std::vector< Creature * > zoo_ptr_

  Указатели на текущее состояние поля
- std::vector< Creature * > empty_zoo_ptr_

  Указатели на запасное состояние поля
- std::vector< Creature > storage_

  Хранилище существ
- std::vector< unsigned int > colors_

  Хранит цвета каждого существа, используется при создании текстуры.
- GLuint texture_

  Указатель на текстуру

Объявления и описания членов классов находятся в файлах:

- objects/scenario/include/scenario/scenario.h
- objects/scenario/scenario.cpp

7.6 Класс cellworld::UI

Открытые члены

- void updateWindowSize (int width, int height)
- void loadScene ()

Объявления и описания членов классов находятся в файлах:

- objects/UI/include/UI/UI.h
- objects/UI/UI.cpp

7.7 Класс cellworld::WindowTemplates

Статические открытые данные

- static const ImGuiWindowFlags invisibleWindow
- static const ImGuiWindowFlags scrollBarOnly
- static const ImGuiWindowFlags menuBar

7.7.1 Данные класса

7.7.1.1 invisibleWindow

const ImGuiWindowFlags cellworld::WindowTemplates::invisibleWindow [static]

Инициализатор

```
= ImGuiWindowFlags_NoTitleBar | ImGuiWindowFlags_NoScrollbar | ImGuiWindowFlags_NoMove
        | ImGuiWindowFlags_NoResize | ImGuiWindowFlags_NoBackground | ImGuiWindowFlags_NoCollapse |
    ImGuiWindowFlags_AlwaysAutoResize
```

7.7.1.2 menuBar

const ImGuiWindowFlags cellworld::WindowTemplates::menuBar [static]

Инициализатор

```
= ImGuiWindowFlags_NoTitleBar | ImGuiWindowFlags_NoScrollbar | ImGuiWindowFlags_NoMove
        | ImGuiWindowFlags_NoResize | ImGuiWindowFlags_NoBackground | ImGuiWindowFlags_NoCollapse |
    ImGuiWindowFlags_AlwaysAutoResize | ImGuiWindowFlags_MenuBar
```

7.7.1.3 scrollBarOnly

const ImGuiWindowFlags cellworld::WindowTemplates::scrollBarOnly [static]

Инициализатор

```
= ImGuiWindowFlags_NoTitleBar | ImGuiWindowFlags_NoMove | ImGuiWindowFlags_NoResize
        | ImGuiWindowFlags_NoBackground | ImGuiWindowFlags_NoCollapse | ImGuiWindowFlags_AlwaysAutoResize
```

Объявления и описания членов класса находятся в файле:

- objects/UI/include/UI/UI.h

Глава 8

Файлы

8.1 Файл objects/creature/include/creature/creature.h

Ядро программы, описано поведение мира.
#include <Eigen/Core>
#include <glad/glad.h>
#include <omp.h>
#include <array>
#include <fstream>
#include <random>
#include <vector>

Классы

- struct cellworld::Genome

  Хранит геном существа.
- class cellworld::Creature

  Класс существа.
- class cellworld::Field

  Класс Поля, хранит существ.

Определения типов

- using cellworld::NeuronNetwork = Eigen::Matrix< float, output_neurons_count,(4 *look_input_count+input_neu

  Хранит веса нейросети существа
- using cellworld::Position = std::pair< int, int >

Перечисления

- enum cellworld::State { cellworld::not_exist , cellworld::dead , cellworld::alive }

  Все возможные состояния существа.
- enum Directions { up , down , left , right }
- enum cellworld::Coefficients {
  cellworld::mass_into_energy , cellworld::mass_capacity , cellworld::starting_energy , cellworld::mass_cost
  ,
  cellworld::change_speed_module_cost , cellworld::braking_force , cellworld::mutation_strength ,
  cellworld::coefficients_count }

  Набор коэффициентов, которые определяют правила мира.
- enum cellworld::InputNeurons {
  cellworld::pos_x , cellworld::pos_y , cellworld::speed_module , cellworld::energy ,
  cellworld::bias , cellworld::input_neurons_count }

  Основные входные нейроны существа Содержит информацию о внутреннем состоянии существа

- enum cellworld::OutputNeurons {
  cellworld::change_speed_module , cellworld::vertical_or_horizontal , cellworld::decrease_or_increase , cellworld::reproduce ,
  cellworld::output_neurons_count }

  Выходные нейроны существа
- enum cellworld::LookInput {
  cellworld::distance , cellworld::color_red , cellworld::color_green , cellworld::color_blue ,
  cellworld::look_input_count }

  Дополнительные входные нейроны существа.

Функции

- std::uniform_real_distribution< float > cellworld::dis (-4.0f, 4.0f)

  Генератор случайных float в диапозоне [-4,4]. Определяет разброс значений у весов нейросети.

Обработка коллизий (ситуаций, когда 2 сущетсва претендуют на 1 позицию)

- void cellworld::conjoin (Creature *&champion, Creature *&candidate)
- void cellworld::conjoin (Creature &champion, Creature &candidate)

Переменные

- std::mt19937 cellworld::generator_

  Общий детерминированный генератор случайных чисел. Любая генерация случайного числа должна быть сделана с помощью данного генератора(за исключением генерации сида).
- constexpr Position cellworld::bad_position ={-1,-1}

8.1.1 Подробное описание

Ядро программы, описано поведение мира.
Содержит классы существа и поля, логику взаимодействия между ними.

8.1.2 Перечисления

8.1.2.1 State

enum cellworld::State
Все возможные состояния существа.

Элементы перечислений

not_exist	существа нет
dead	мёртвое существо
alive	живое существо

8.2 creature.h

См. документацию.

```
00001
00009 #ifndef EVOLVING_WORLD_2023_Q2
00010 #define EVOLVING_WORLD_2023_Q2
00011 #define EIGEN_NO_DEBUG
00012
00013 #include <Eigen/Core>
00014 #include<glad/glad.h>
00015
00016 #include<omp.h>
00017
00018 #include <array>
00019 #include <fstream>
00020 #include <random>
```

```
00021 #include <vector>
00022
00023 #define GL_SILENCE_DEPRECATION
00024
00025
00026
00027
00028
00029
00030
00031
00032 namespace cellworld
00033 {
00039     inline std::mt19937 generator_;
00040
00047     inline std::uniform_real_distribution<float> dis(-4.0f, 4.0f);
00048
00049
00055     enum State {
00056         not_exist,
00057         dead,
00058         alive,
00059     };
00060
00061
00062     enum Directions {
00063         up,
00064         down,
00065         left,
00066         right,
00067
00068     };
00069
00070
00071
00072
00080     enum Coefficients {
00088         mass_into_energy,
00089
00090
00096         mass_capacity,
00097
00098
00104         starting_energy,
00105
00111         mass_cost,
00112
00113
00120         change_speed_module_cost,
00121
00122
00128         braking_force,
00129
00130
00138         mutation_strength,
00139
00140
00142         coefficients_count
00143
00144     };
00145
00157     enum InputNeurons {
00159         pos_x,
00160
00162         pos_y,
00163
00165         speed_module,
00166
00168         energy,
00169
00171         bias,
00172
00174         input_neurons_count
00175     };
00176
00177
00178
00183     enum OutputNeurons {
00185         change_speed_module,
00186
00188         vertical_or_horizontal,
00189
00191         decrease_or_increase,
00192
00194         reproduce,
00195
00197         output_neurons_count
```

```
00198     };
00199
00200
00208     enum LookInput {
00210         distance,
00212         color_red,
00214         color_green,
00216         color_blue,
00217
00218
00220         look_input_count
00221     };
00222
00223
00225 using NeuronNetwork = Eigen::Matrix<float,output_neurons_count , (4 * look_input_count + input_neurons_count)>;
00226
00228 using Position = std::pair<int,int>;
00229 constexpr Position bad_position={-1,-1};
00230
00231
00232
00241 struct Genome {
00242     Genome() :color(),mass(),neuron_network() {}
00243     explicit Genome(unsigned int in_color):color(in_color),mass(),neuron_network() {}
00244     unsigned int color;
00245     unsigned int mass;
00246     NeuronNetwork neuron_network;
00247 };
00248
00258 class Creature {
00259 public:
00260
00262     static Genome generateGenome();
00263
00265     static void generateGenome(Genome&);
00266
00272     static void mixGen(float& gen1, const float& gen2);
00273
00274
00280     static unsigned int mixGen(const unsigned int& gen1, const unsigned int& gen2);
00281
00285     static Genome createGenome(const Genome& ancestor);
00286
00291     static unsigned int energyColor(int energy);
00292
00293
00295     inline static std::array<float, coefficients_count> coeff_{ 0 };
00296
00298     inline static bool is_breedable = 1;
00299
00301     inline static unsigned int base_color_ = 0xAfAfAfff;
00302
00303
00309
00310
00316     Creature(const Genome& gen, const Position& pos);
00317
00318
00320     Creature();
00321
00322
00327     Creature(float energy, const Position& pos);
00329
00330
00331
00332     Creature(const Creature&)=default;
00333     Creature(Creature&&) = default;
00334     Creature& operator=(const Creature&) = default;
00335     Creature& operator=(Creature&&) = default;
00336
00337
00342
00343
00350     void makeAlive(Creature& ancestor, const Position& pos);
00351
00352
00358     void makeAlive(const Position& pos);
00359
00360
00366     void die();
00367
00368
00374     void stopExisting() { state_ = not_exist; creatures_genome_.color = base_color_; energy_ = 0; speed_module_ = 0;
    }
00376
00377
00381     const int& getState() const { return state_; }
```

```
00382     const int& getDirection() const {return speed_direction_;}
00383     const int& getSpeed() const {return speed_module_;}
00384
00385
00392     int& getX();
00393
00394
00401     int& getY();
00402
00403
00404     const float& getEnergy() const { return energy_; }
00405     const float& getEnergyLimit() const { return energy_limit_; }
00406     const int& getMass() const { return (creatures_genome_.mass); }
00407     const unsigned int& getColor() const { return (creatures_genome_.color); }
00408     unsigned int getBlue() const { return((getColor() » 8) & 0xff); }
00409     unsigned int getGreen() const { return((getColor() » 16) & 0xff); }
00410     unsigned int getRed() const { return((getColor() » 24) & 0xff); }
00411     const Genome& getGenome() { return creatures_genome_; }
00413
00414
00416     float Leftover();
00417
00418
00420     bool wantToReproduce()const {return (output_neurons_.size()!=0) && is_breedable &&
      (output_neurons_.coeff(reproduce)>0); }
00421
00422
00429     void look(Creature& found, int direction);
00430
00431
00437     void getInfo();
00438
00439
00445     void reverseInput();
00446
00447
00453     void think();
00454
00462     void act();
00463
00467     void eat(Creature&);
00471     void addEnergy(const float& energy);
00472
00473
00475     void buildIO();
00476
00477
00478
00484     int pos_x_;
00485     int pos_y_;
00487
00488
00489
00490 private:
00491
00492
00494     int state_;
00495
00497     float energy_;
00498
00500     float energy_limit_;
00501
00503     int speed_module_;
00504
00506     int speed_direction_;
00507
00509     Genome creatures_genome_;
00510
00512     Eigen::MatrixXf input_neurons_;
00513
00515     Eigen::MatrixXf output_neurons_;
00516
00518     std::tuple<int,int,float> tmp_;
00519
00520 };
00521
00535
00539 void conjoin(Creature*& master, Creature*& candidate);
00544 void conjoin(Creature& master, Creature& candidate);
00546
00547
00548
00549
00550
00562 class Field{
00563
00564 public:
```

```
00570     Field(int size_x=0, int size_y=0);
00571
00572
00584     void updatePositions();
00585
00586
00594     void updateStates();
00595
00596
00601     const Creature& getCreature(const Position& pos) const;
00602     Creature& getCreature(const Position& pos);
00603
00604     const Creature& getCreature(const int& index) const { return *zoo_ptr_[index]; }
00605     Creature& getCreature(const int& index) { return *zoo_ptr_[index]; }
00606
00607     const Creature& operator[](const int& index) const {return *zoo_ptr_[index];}
00608     Creature& operator[](const int& index) { return *zoo_ptr_[index]; }
00610
00611
00612
00616     unsigned int getColor(const Position& pos)const {return getCreature(pos).getColor();}
00617     unsigned int getColor(const int& index) const { return getCreature(index).getColor(); }
00618     int sizeX() const { return size_x_; }
00619     int sizeY() const { return size_y_; }
00620     int size() const { return size_; }
00622
00626     bool validX(const int& x) const;
00627     bool validY(const int& y) const;
00629
00633     Position findClosePosition(Creature* ancestor);
00634
00639     Creature& findCreature(Creature* finder, int direction);
00640
00641
00643     void clear();
00644
00645
00651     void createTexture();
00653     void* getTexture() { return (void*)(texture_); }
00655     const GLuint& getGLTexture() {return texture_;}
00657     void updateTexture();
00659     void unbindTexture();
00661
00662
00663     inline static Creature bad_creature=Creature();
00664
00665
00666 private:
00668     int size_x_;
00670     int size_y_;
00672     int size_;
00673
00674 protected:
00676     std::vector<Creature*> zoo_ptr_;
00678     std::vector<Creature*> empty_zoo_ptr_;
00680     std::vector<Creature> storage_;
00682     std::vector<unsigned int> colors_;
00684     GLuint texture_;
00685
00686 };
00687
00688
00697 };
00698
00699 #endif
```

8.3 Файл objects/save_system/include/save_system/save_system.h

Сиситема сохранений.
#include <fstream>
#include <vector>
#include <string>
#include <cstdio>
#include <utility>

Классы

- class cellworld::FileSystem

Класс хранения файлов.

8.3.1 Подробное описание

Сиситема сохранений.

8.4 save_system.h

См. документацию.

```
00001
00002 #ifndef EVOLVING_WORLD_2023_Q2_FILESYSYTEM
00003 #define EVOLVING_WORLD_2023_Q2_FILESYSYTEM
00004
00005
00011 #include<fstream>
00012 #include<vector>
00013 #include<string>
00014 #include<cstdio>
00015 #include<utility>
00016
00017 namespace cellworld {
00018
00019
00020
00021
00034     class FileSystem {
00035     public:
00037         FileSystem(std::string name);
00039         FileSystem();
00040
00042         void addFileName(const char* file_name);
00043
00045         void removeFileName(int index);
00046
00048         void saveFileNames();
00050         void loadFileNames();
00051
00052
00054         void checkFileNames();
00055
00057         bool findFileName(const char* file_name);
00058
00060         std::string getValidFileName(std::string file_name);
00061
00063         const std::vector<std::string>& getFileNames() { return files_; }
00064     private:
00066         void checkFileName(int index);
00067
00069         std::string store_names_file_;
00070
00072         std::vector<std::string> files_;
00073     };
00075 }
00076
00077 #endif
```

8.5 scenario.h

```
00001
00002
00003 #ifndef EVOLVING_WORLD_2023_Q2_SCENARIO
00004 #define EVOLVING_WORLD_2023_Q2_SCENARIO
00005
00006 #include<creature/creature.h>
00007 #include"imgui.h"
00008 #include<array>
00009 #include<vector>
00010 #include<random>
00011 #include<omp.h>
00012 namespace cellworld {
00013
00014
00015
00016
00017     class Scenario: public Field {
00018         friend void saveWorld(const char* path, Scenario* current_field, unsigned int seed);
00019         friend void loadWorld(const char* path, Scenario* current_field, unsigned int& seed);
00020     public:
00021         Scenario();
00022         Scenario(int size_x, int size_y);
```

```
00023
00024
00025         void makeNew();
00026         void makeOneStep();
00027
00028         void spawnCreatures(int amount);//использовать только на пустом поле
00029
00030         void makeRewards(Position,Position, float strength);//за нахожение на прямоугольнике, заданный двумя
      позициями, существо каждый ход получает энергию=strength
00031         void giveRewards();
00032         void CancelRewardsChange() { std::swap(rewards_, rewards_backup_); }
00033         void resetRewards();
00034
00035         void rewardsEditor(ImVec2 window_pos, ImVec2 window_size, float strength, ImTextureID texture);
00036         void updateRewardsTexture();
00037
00038         void newCycle();
00039
00040         int getInitialPopulation(){return initial_population_;}
00041         long long getIteration(){return iteration_;}
00042         int cycle_len_;
00043     private:
00044         Position convertInput(ImVec2 begin, ImVec2 input, int square_size);
00045
00046         int initial_population_;
00047
00048         long long iteration_;
00049         std::vector<float> rewards_;
00050         std::vector<float> rewards_backup_;
00051         std::vector<Creature*> survivors_;
00052         std::vector<int> positions_;
00053
00054     };
00055
00056
00057 }
00058
00059 #endif
```

8.6 UI.h

```
00001
00002
00003 #ifndef EVOLVING_WORLD_2023_Q2_UI
00004 #define EVOLVING_WORLD_2023_Q2_UI
00005
00006 #include "imgui.h"
00007 #include<creature/creature.h>
00008 #include<save_system/save_system.h>
00009 #include<scenario/scenario.h>
00010 #include "imgui_stdlib.h"
00011 namespace cellworld {
00012
00013     enum Scene {
00014         start_screen,
00015         creation_of_the_world,
00016         load_world_screen,
00017         simulation_of_the_world,
00018     };
00019
00020
00021     class WindowTemplates {
00022     public:
00023         static const ImGuiWindowFlags invisibleWindow = ImGuiWindowFlags_NoTitleBar |
      ImGuiWindowFlags_NoScrollbar | ImGuiWindowFlags_NoMove
00024             | ImGuiWindowFlags_NoResize | ImGuiWindowFlags_NoBackground | ImGuiWindowFlags_NoCollapse |
      ImGuiWindowFlags_AlwaysAutoResize;
00025         static const ImGuiWindowFlags scrollBarOnly = ImGuiWindowFlags_NoTitleBar | ImGuiWindowFlags_NoMove |
      ImGuiWindowFlags_NoResize
00026             | ImGuiWindowFlags_NoBackground | ImGuiWindowFlags_NoCollapse | ImGuiWindowFlags_AlwaysAutoResize;
00027         static const ImGuiWindowFlags menuBar = ImGuiWindowFlags_NoTitleBar | ImGuiWindowFlags_NoScrollbar |
      ImGuiWindowFlags_NoMove
00028             | ImGuiWindowFlags_NoResize | ImGuiWindowFlags_NoBackground | ImGuiWindowFlags_NoCollapse |
      ImGuiWindowFlags_AlwaysAutoResize | ImGuiWindowFlags_MenuBar;
00029     };
00030
00031
00032     static void HelpMarker(const char* desc, int width)
00033     {
00034         ImGui::SameLine();
00035         ImGui::TextDisabled("(?)");
00036         if (ImGui::IsItemHovered(ImGuiHoveredFlags_DelayShort))
00037         {
00038             ImGui::BeginTooltip();
00039             ImGui::SetWindowFontScale(width / 4096.0f);
```

```
00040             ImGui::PushTextWrapPos(ImGui::GetFontSize() * 35);
00041             ImGui::TextUnformatted(desc);
00042             ImGui::PopTextWrapPos();
00043             ImGui::EndTooltip();
00044         }
00045     }
00046 
00047 
00048     class UI {
00049     public:
00050         UI() : scene_(0), previous_scene_(0), scene_is_changed_(0), width_(1), height_(1), seed_(0), scenario_(100,50) {}
00051         void updateWindowSize(int width, int height){ width_ =width; height_= height;}
00052         void loadScene();
00053 
00054     private:
00055         void sceneUpdate(int scene);
00056         void startScreen();
00057         void CreationOfTheWorld();
00058         void loadWorldScreen();
00059         void SimulationOfTheWorld();
00060 
00061         int scene_;
00062         int previous_scene_;
00063         bool scene_is_changed_;
00064 
00065         int width_;
00066         int height_;
00067 
00068         FileSystem file_names_;
00069         Scenario scenario_;
00070         unsigned int seed_;
00071     };
00072 }
00073 
00074 #endif
```

Предметный указатель