PacManClone

AUTHOR
Версия
Ср Май 18 2022

Съдържание

Table of contents

Именни пространства Указател

Именни пространства Списък

пълен списък с именни пространства с кратко описание:

Класове Указател

Класове Списък

Класове, структури, обединения и интерфейси с кратко описание:

Файлове Списък

Файлове Списък

Пълен списък с файлове с кратко описание:

Именни пространства Документация

PacMan Именно пространството Справка

Класове

class **Game**
class **Ghost**
class **Map**
class **Player**
class **Point**

Класове Документация

PacMan::Game Клас Препратка

`#include <PacMan.h>`

Общодостъпни членове функции

Game ()=default
void **Init** ()
void **Update** ()

Конструктор & Деструктор Документация

PacMan::Game::Game () `[default]`

Членове Функции(методи) Документация

void PacMan::Game::Init ()

void PacMan::Game::Update ()

Документация за клас генериран от следните файлове:

C:/Users/Krusto/source/repos/PacManClone/PacManClone/**PacMan.h**
C:/Users/Krusto/source/repos/PacManClone/PacManClone/**PacMan.cpp**

PacMan::Ghost Клас Препратка

`#include <Ghost.h>`

Общодостъпни членове функции

Ghost ()=default
void **Move** (**Point** dir)
float **GetTargetDistance** (uint32_t **direction**)
bool **IsCollidingPacman** (**Player** &player)
void **UpdateTarget** (uint32_t pacmanDirection, **Point** ghostPosition, **Point** pacmanPosition, **Map**< 21, 21 > &map)
void **Update** (**Map**< 21, 21 > &map, **Player** &player, **Ghost** &ghost)

Общодостъпни атрибути

uint32_t **id** {}
uint32_t **frightenedSpeedTimer** {}
bool **frightenedMode** {}
uint32_t **movementMode** {0}
Point position {}
Point home {10,10}
uint32_t **direction** {}
Point target {}

Конструктор & Деструктор Документация

PacMan::Ghost::Ghost () `[default]`

Членове Функции(методи) Документация

float PacMan::Ghost::GetTargetDistance (uint32_t *direction*) `[inline]`

Функция която връща разстоянието до целта

bool PacMan::Ghost::IsCollidingPacman (Player & *player*) `[inline]`

Функция която проверява дали има колизия м-у духчето и пакман

void PacMan::Ghost::Move (Point *dir*) `[inline]`

Функция за движение на духчето в някаква посока

void PacMan::Ghost::Update (Map< 21, 21 > & *map*, Player & *player*, Ghost & *ghost*) `[inline]`

Функция за опресняване на духчето

void PacMan::Ghost::UpdateTarget (uint32_t *pacmanDirection*, Point *ghostPosition*, Point *pacmanPosition*, Map< 21, 21 > & *map*) `[inline]`

Функция която опреснява целта

Член данни Документация

uint32_t PacMan::Ghost::direction {}

Посока на духчето

bool PacMan::Ghost::frightenedMode {}

Променлива която служи за проверка дали е уплашено

uint32_t PacMan::Ghost::frightenedSpeedTimer {}

Таймер за колко време се страхува

Point PacMan::Ghost::home {10,10}

Позиция на къщата на духчето

uint32_t PacMan::Ghost::id {}

Идентификатор на духчето

uint32_t PacMan::Ghost::movementMode {0}

Режим на движение

Point PacMan::Ghost::position {}

Позижия на духчето

Point PacMan::Ghost::target {}

Цел на духчето

Документация за клас генериран от следният файл:

C:/Users/Krusto/source/repos/PacManClone/PacManClone/**Ghost.h**

PacMan::Map< SizeX, SizeY > Клас Шаблон Препратка

`#include <Map.h>`

Общодостъпни членове функции

Map ()=default
void **Set** (size_t x, size_t y, int8_t value)
void **Set** (**Point** pos, int8_t value)
int8_t **Get** (size_t x, size_t y) const
int8_t **Get** (**Point** pos) const
bool **IsValid** (**Point** pos)
auto & **data** ()
auto **GetSizeX** () const
auto **GetSizeY** () const
const void **Draw** (**Point** playerPosition, **Point** ghostPosition[4], bool frightened[4]) const
void **Fill** (int8_t symbol)
void **Fill** (std::string **data**[SizeY])
bool **isColliding** (**Point** p)

Конструктор & Деструктор Документация

template<uint32_t SizeX, uint32_t SizeY> PacMan::Map< SizeX, SizeY >::Map () [default]

Конструктор по подразбиране

Членове Функции(методи) Документация

template<uint32_t SizeX, uint32_t SizeY> auto & PacMan::Map< SizeX, SizeY >::data () [inline]

Функция която връща низа с информация на картата

template<uint32_t SizeX, uint32_t SizeY> const void PacMan::Map< SizeX, SizeY >::Draw (Point *playerPosition*, Point *ghostPosition*[4], bool *frightened*[4]) const

Функция която рисува

template<uint32_t SizeX, uint32_t SizeY> void PacMan::Map< SizeX, SizeY >::Fill (int8_t *symbol*)

Функция която запълва картата с някакъв символ

template<uint32_t SizeX, uint32_t SizeY> void PacMan::Map< SizeX, SizeY >::Fill (std::string *data*[SizeY])

template<uint32_t SizeX, uint32_t SizeY> int8_t PacMan::Map< SizeX, SizeY >::Get (Point *pos*) const [inline]

Функция която връща стойност на клетка с позиция pos

template<uint32_t SizeX, uint32_t SizeY> int8_t PacMan::Map< SizeX, SizeY >::Get (size_t *x*, size_t *y*) const [inline]

Функция която връща стойност на клетка с позиция x,y

template<uint32_t SizeX, uint32_t SizeY> auto PacMan::Map< SizeX, SizeY >::GetSizeX () const `[inline]`

Функция която връща широчина на картата

template<uint32_t SizeX, uint32_t SizeY> auto PacMan::Map< SizeX, SizeY >::GetSizeY () const `[inline]`

Функция която връща височина на картата

template<uint32_t SizeX, uint32_t SizeY> bool PacMan::Map< SizeX, SizeY >::isColliding (Point *p*) `[inline]`

Функция която проверява дали позицията се намира на мястото на стена

template<uint32_t SizeX, uint32_t SizeY> bool PacMan::Map< SizeX, SizeY >::IsValid (Point *pos*) `[inline]`

Функция която проверява дали точката е в картата

template<uint32_t SizeX, uint32_t SizeY> void PacMan::Map< SizeX, SizeY >::Set (Point *pos*, int8_t *value*) `[inline]`

Функция която задава стойност на клетка с позиция pos

template<uint32_t SizeX, uint32_t SizeY> void PacMan::Map< SizeX, SizeY >::Set (size_t *x*, size_t *y*, int8_t *value*) `[inline]`

Функция която задава стойност на клетка с позиция x,y

Документация за клас генериран от следните файлове:

C:/Users/Krusto/source/repos/PacManClone/PacManClone/**Map.h**
C:/Users/Krusto/source/repos/PacManClone/PacManClone/**Map.cpp**

PacMan::Player Клас Препратка

`#include <Player.h>`

Общодостъпни членове функции

Player ()=default
Player (const **Point** startPosition)
void **Move** (**Point** dir)
void **Update** (**Map**< 21, 21 > &map, **Point** ghostPositions[4])

Общодостъпни атрибути

bool **isDead** {}
Point position {}
uint32_t **direction** {}
uint32_t **score** {}
uint32_t **scoreMultiplier** {}
uint32_t **multiplierTicks** {}
bool **isPowered** {}

Подробно описание

Клас Играч

Конструктор & Деструктор Документация

PacMan::Player::Player () `[default]`

Конструктор по подразбиране

PacMan::Player::Player (const Point *startPosition*) `[inline]`

Конструктор при който се задава началната позиция на играча

Членове Функции(методи) Документация

void PacMan::Player::Move (Point *dir*) `[inline]`

Функция която мърда играча в дадена посока

void PacMan::Player::Update (Map< 21, 21 > & *map*, Point *ghostPositions*[4])

Функция която опреснява играча и картата

Член данни Документация

uint32_t PacMan::Player::direction {}

Посока в която се движи играча

bool PacMan::Player::isDead {}

Флаг който служи за проверка дали играчът е умрял

bool PacMan::Player::isPowered {}

Флаг за проверка дали може да изяде духче

uint32_t PacMan::Player::multiplierTicks {}

Изминали тикове

Point PacMan::Player::position {}

Позиция на играча

uint32_t PacMan::Player::score {}

Точки които е събрал играча

uint32_t PacMan::Player::scoreMultiplier {}

умножител на точки

Документация за клас генериран от следните файлове:

C:/Users/Krusto/source/repos/PacManClone/PacManClone/**Player.h**
C:/Users/Krusto/source/repos/PacManClone/PacManClone/**Player.cpp**

PacMan::Point Клас Препратка

```
#include <Point.h>
```

Общодостъпни членове функции

Point ()=default
Point (float **x**, float **y**)
Point (const **Point** &)=default
Point & **operator=** (const **Point** &other)=default
bool **operator==** (const **Point** &rhs)
Point & **operator+=** (const **Point** &rhs)
Point & **operator-=** (const **Point** &rhs)
bool **operator!=** (const **Point** &rhs)
Point operator+ (const **Point** &rhs)
Point & **operator*** (float rhs)
Point GetTrunc ()

Общодостъпни атрибути

float **x** {}
float **y** {}

Конструктор & Деструктор Документация

PacMan::Point::Point () `[default]`

Конструктор по подразбиране

PacMan::Point::Point (float *x*, float *y*) `[inline]`

Конструктор който взима x,y като параметри

PacMan::Point::Point (const Point &) `[default]`

Конструктор който копира данните от друга точка

Членове Функции(методи) Документация

Point PacMan::Point::GetTrunc () `[inline]`

bool PacMan::Point::operator!= (const Point & *rhs*) `[inline]`

Point & PacMan::Point::operator* (float *rhs*) `[inline]`

Point PacMan::Point::operator+ (const Point & *rhs*) `[inline]`

Point & PacMan::Point::operator+= (const Point & *rhs*) `[inline]`

Point & PacMan::Point::operator-= (const Point & *rhs*) `[inline]`

Point & PacMan::Point::operator= (const Point & *other*) `[default]`

bool PacMan::Point::operator== (const Point & *rhs*) `[inline]`

Член данни Документация

float PacMan::Point::x {}

float PacMan::Point::y {}

Документация за клас генериран от следният файл:
C:/Users/Krusto/source/repos/PacManClone/PacManClone/**Point.h**

Файлове Документация

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Ghost.h Файл Справка

```
#include "Point.h"
#include <iostream>
#include <stdlib.h>
#include "Map.h"
#include "Math.h"
#include "Player.h"
```

Класове

class **PacMan::Ghost**

Именни пространства

namespace **PacMan**

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Ghost.h

Вижте документацията за този файл.1 #pragma once
2 #include "Point.h"
3 #include <iostream>
4 #include <stdlib.h>
5 #include "Map.h"
6 #include "Math.h"
7 #include "Player.h"
8
9 namespace PacMan {
10 class Ghost {
11 public:
12 Ghost() = default;
13
17 void Move(Point dir) { position += dir; }
18
22 float GetTargetDistance(uint32_t direction) {
23 float x = position.x;
24 float y = position.y;
25
26 switch (direction)
27 {
28 case 0:
29 {
30 x += 1;
31
32 break;
33 }
34 case 1:
35 {
36 y -= 1;
37
38 break;
39 }
40 case 2:
41 {
42 x -= 1;
43
44 break;
45 }
46 case 3:
47 {
48 y += 1;
49 }
50 }
51 return GetDistance({ x,y }, target);
52 }
53
57 bool IsCollidingPacman(Player& player) {
58 return player.position.GetTrunc() == position;
59 }
63 void UpdateTarget(uint32_t pacmanDirection, Point ghostPosition, Point
pacmanPosition,Map<21,21>& map) {
64 if (movementMode == 0) // scatter mode
65 {
66 //отива в края на картата
67 switch (id)
68 {
69 case 0:
70 {
71 target = {(map.GetSizeX() - 1.0f), 0.0f };
72
73 break;
74 }
75 case 1:
76 {
77 target = { 0.0f, 0.0f };
78
79 break;
80 }
81 case 2:
82 {

```
83                                  target = { (map.GetSizeX() - 1.0f), (map.GetSizeY()
- 1.0f) };
84
85                                  break;
86                             }
87                             case 3:
88                             {
89                                  target = { 0.0f,(map.GetSizeY() - 1.0f) };
90                             }
91                             }
92                        }
93                        else //chase mode
94                        {
95                             switch (id)
96                             {
97                             case 0: //червеният дух нацелва пакман
98                             {
99                                  target = pacmanPosition;
100
101                                  break;
102                             }
103                             case 1: //розовият дух нацелва 4 клетки пред пакман
104                             {
105                                  target = pacmanPosition;
106
107                                  switch (pacmanDirection)
108                                  {
109                                  case 0:
110                                  {
111                                       target += {2, 0};
112
113                                       break;
114                                  }
115                                  case 1:
116                                  {
117                                       target -= {0, 2};
118
119                                       break;
120                                  }
121                                  case 2:
122                                  {
123                                       target -= {2, 0};
124                                       break;
125                                  }
126                                  case 3:
127                                  {
128                                       target += {0, 2};
129
130                                  }
131                                  }
132
133                                  break;
134                             }
135                             case 2: // синият нацелва 2 клетки пред пакман
136                             {
137                                  target = pacmanPosition;
138
139                                  switch (pacmanDirection)
140                                  {
141                                  case 0:
142                                  {
143                                       target += {1, 0};
144
145                                       break;
146                                  }
147                                  case 1:
148                                  {
149                                       target -= {0, 1};
150
151                                       break;
152                                  }
153                                  case 2:
154                                  {
155                                       target -= {1, 0};
156
157                                       break;
158                                  }
```

```
159                                 case 3:
160                                 {
161                                      target += {0, 1};
162                                 }
163                                 }
164
165                                 target += {target.x - ghostPosition.x,
166                                                 target.y - ghostPosition.y};
167
168                                 break;
169                            }
170                            case 3: // оранжевият дух преследва пакман и след това
отива в scatter режим
171                            {
172
173                                 if ( GetDistance(pacmanPosition,position) >= 4)
174                                 {
175                                      target = pacmanPosition;
176                                 }
177                                 else
178                                 {
179                                      target = { 0.0f, (map.GetSizeY() - 1.0f) };
180                                 }
181                            }
182                            }
183                       }
184            }
188            void Update(Map<21, 21>& map, Player& player,Ghost& ghost) {
189                 bool move = 0;
190
191                 unsigned char available_ways = 0;
192                 float speed = 1;
193
194                 bool walls[4]{};
195
196                 UpdateTarget(player.direction,ghost.position,
player.position,map);
197
198                 walls[0] = map.Get(position + Point{ speed,0 }) == '#';
199                 walls[1] = map.Get(position + Point{ 0,-speed }) == '#';
200                 walls[2] = map.Get(position + Point{ -speed,0 }) == '#';
201                 walls[3] = map.Get(position + Point{ 0,speed }) == '#';
202
203                 if (frightenedMode != 1)
204                 {
205                      unsigned char optimal_direction = 4;
206
207                      // духчето може да мърда
208                      move = 1;
209
210                      for (unsigned char a = 0; a < 4; a++)
211                      {
212                           //духчетата не могат да се обърнат, стига да не е нужно
213                           if (a == (2 + direction) % 4)
214                           {
215                                continue;
216                           }
217                           else if (walls[a] == false)
218                           {
219                                if (optimal_direction == 4)
220                                {
221                                     optimal_direction = a;
222                                }
223
224                                available_ways++;
225
226                                if (GetTargetDistance(a) <
GetTargetDistance(optimal_direction))
227                                {
228                                     // оптималната посока за да стигне целта
229                                     optimal_direction = a;
230                                }
231                           }
232                      }
233
234                      if (available_ways > 1)
235                      {
```

```
236                         // когато стигне кръстопът избира най добрата посока
237                         direction = optimal_direction;
238                    }
239                    else
240                    {
241                         if (optimal_direction == 4)
242                         {
243                              // обръща се духчето на обратно защото трябва
244                              direction = (2 + direction) % 4;
245                         }
246                         else
247                         {
248                              direction = optimal_direction;
249                         }
250                    }
251               }
252               else
253               {
254                    unsigned char random_direction = rand() % 4;
255
256                    if (frightenedSpeedTimer == 0)
257                    {
258
259                         move = 1;
260
261                         frightenedSpeedTimer = 30;
262
263                         for (unsigned char a = 0; a < 4; a++)
264                         {
265                              // не може да се обръщат дори и да се страхуват
266                              if (a == (2 + direction) % 4)
267                              {
268                                   continue;
269                              }
270                              else if (walls[a] == 0)
271                              {
272                                   available_ways++;
273                              }
274                         }
275
276                         if (available_ways > 0)
277                         {
278                              while (walls[random_direction] == 1 ||
random_direction == (2 + direction) % 4)
279                              {
280                                   // взимаме произволна посока докато не може да
я използваме
281                                   random_direction = rand() % 4;
282                              }
283
284                              direction = random_direction;
285                         }
286                         else
287                         {
288                              // ако няма друг път се обръща наобратно
289                              direction = (2 + direction) % 4;
290                         }
291                    }
292                    else
293                    {
294                         frightenedSpeedTimer--;
295                    }
296               }
297               // ако може да се мръдне го местим
298               if (move)
299               {
300                    switch (direction)
301                    {
302                    case 0:
303                    {
304                         position.x += speed;
305
306                         break;
307                    }
308                    case 1:
309                    {
310                         position.y -= speed;
```

```
311
312                         break;
313                     }
314                     case 2:
315                     {
316                         position.x -= speed;
317
318                         break;
319                     }
320                     case 3:
321                     {
322                         position.y += speed;
323                     }
324                     }
325
326                     // когато мине през портала го пращаме от другата страна
327                     if (position.x <= -1)
328                     {
329                         position.x = map.GetSizeX() - speed;
330                     }
331                     else if (position.x >= map.GetSizeX())
332                     {
333                         position.x = speed;
334                     }
335                 }
336
337                 if (position.GetTrunc() == player.position.GetTrunc())
338                 {
339                     if (frightenedMode == 0) // когато не е уплашен и докосне
пакман, играта приключва
340                     {
341                         player.isDead = 1;
342                     }
343                     else //иначе бяга към клетката
344                     {
345                         frightenedMode = 2;
346
347                         target = home;
348                     }
349                 }
350             }
354             uint32_t id{};
358             uint32_t frightenedSpeedTimer{};
362             bool frightenedMode{};
366             uint32_t movementMode{0};
370             Point position{};
374             Point home{10,10};
378             uint32_t direction{};
382             Point target{};
383         };
384 }
```

C:/Users/Krusto/source/repos/PacManClone/PacManClone/main.cpp Файл Справка

```
#include <iostream>
#include <Windows.h>
#include "PacMan.h"
#include "WindowsFunctions.h"
```

Функции

int **main** ()

Функции Документация

int main ()

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Map.cpp Файл Справка

```
#include "Map.h"
#include <iostream>
#include <Windows.h>
#include "WindowsFunctions.h"
```

Именни пространства

namespace **PacMan**

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Map.h Файл Справка

```
#include <cstdint>
#include <string>
#include "Point.h"
```

Класове

class **PacMan::Map< SizeX, SizeY >**

Именни пространства

namespace **PacMan**

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Map.h

Вижте документацията за този файл.

```
1  #pragma once
2  #include <cstdint>
3  #include <string>
4  #include "Point.h"
5
6  namespace PacMan {
7
8        template <uint32_t SizeX, uint32_t SizeY>
9        class Map {
10        public:
14             Map() = default;
15
19             void Set(size_t x, size_t y, int8_t value) { m_Data[x][y] = value; }
23             void Set(Point pos, int8_t value) { m_Data[(int)pos.x][(int)pos.y] =
value; }
27             int8_t Get(size_t x, size_t y) const { return m_Data[x][y]; }
31             int8_t Get(Point pos) const {
32                  if (pos.y < height && pos.x < width && pos.x >= 0 && pos.y >= 0) {
33                       return m_Data[(size_t)pos.x][(size_t)pos.y];
34                  }
35                  return INT8_MIN;
36             }
37
41             bool IsValid(Point pos) { return (pos.y < height && pos.x < width &&
pos.x >= 0 && pos.y >= 0); }
45             auto& data() { return &m_Data; }
46
50             auto GetSizeX() const { return SizeX; };
54             auto GetSizeY() const { return SizeY; };
58             const void Draw(Point playerPosition,Point ghostPosition[4], bool
frightened[4]) const;
62             void Fill(int8_t symbol);
63
64             /*
65             * Функция която запълва картата с някакъв низ от редове
66             */
67             void Fill(std::string data[SizeY]);
68
72             bool isColliding(Point p) {
73                  char c = m_Data[(int)p.GetTrunc().x][(int)p.GetTrunc().y];
74                  return c == '#';
75             }
76
77       private:
81             char m_Data[SizeX][SizeY]{};
85             uint32_t width = SizeX;
89             uint32_t height = SizeY;
90       };
91
92 }
```

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Math.cpp Файл Справка

```
#include "Math.h"
```

Функции

float **GetDistance** (**PacMan::Point** p1, **PacMan::Point** p2)

Функции Документация

float GetDistance (PacMan::Point *p1*, PacMan::Point *p2*)

Функция за изчисляване на дистанция използвайки теоремата на Питагор

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Math.h Файл Справка

```
#include "Point.h"
```

Функции

float **GetDistance** (**PacMan::Point** p1, **PacMan::Point** p2)

Функции Документация

float GetDistance (PacMan::Point *p1*, PacMan::Point *p2*)

Функция за изчисляване на дистанция използвайки теоремата на Питагор

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Math.h

Вижте документацията за този файл.

```
1 #pragma once
2 #include "Point.h"
6 float GetDistance(PacMan::Point p1, PacMan::Point p2);
```

C:/Users/Krusto/source/repos/PacManClone/PacManClone/PacMan.cpp Файл Справка

```
#include "PacMan.h"
#include "Map.cpp"
#include <Windows.h>
#include "WindowsFunctions.h"
```

C:/Users/Krusto/source/repos/PacManClone/PacManClone/PacMan.h Файл Справка

```
#include "Player.h"
#include "Map.h"
#include "Ghost.h"
```

Класове

class **PacMan::Game**

Именни пространства

namespace **PacMan**

C:/Users/Krusto/source/repos/PacManClone/PacManClone/PacMan.h

Вижте документацията за този файл.1 #pragma once
2 #include "Player.h"
3 #include "Map.h"
4 #include "Ghost.h"
5
6 namespace PacMan {
7 class Game {
8 public:
9 /*
10 * Конструктор по подразбиране
11 */
12 Game() = default;
13 /*
14 * Функция за инициализация на играта
15 */
16 void Init();
17 /*
18 * Функция която опреснява играта
19 */
20 void Update();
21 private:
22 /*
23 * Предишна позиция на играча
24 */
25 Point m_OldPlayerPosition{};
26 /*
27 * Играч
28 */
29 Player m_Player{};
30 /*
31 * Низ от духчета които се движат на картата
32 */
33 Ghost m_Enemy[4]{};
34 /*
35 * Карта с размери [21,21]
36 */
37 Map<21,21> m_Map{};
38 /*
39 * брой изминали тикове
40 */
41 uint32_t m_Ticks{};
42 };
43 }

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Player.cpp Файл Справка

```
#include "Player.h"
#include "WindowsFunctions.h"
```

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Player.h Файл Справка

```
#include "Point.h"
#include "Map.h"
#include <cstdlib>
```

Класове

class **PacMan::Player**

Именни пространства

namespace **PacMan**

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Player.h

Вижте документацията за този файл.

```
1  #pragma once
2  #include "Point.h"
3  #include "Map.h"
4  #include <cstdlib>
5  namespace PacMan {
9        class Player {
10        public:
14              Player() = default;
18              Player(const Point startPosition) : position(startPosition) {}
22              void Move(Point dir) { position += dir; };
26              void Update(Map<21, 21>& map,Point ghostPositions[4]);
27
31              bool isDead{};
35              Point position{};
39              uint32_t direction{};
43              uint32_t score{};
47              uint32_t scoreMultiplier{};
51              uint32_t multiplierTicks{};
55              bool isPowered{};
56        };
57 }
```

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Point.h Файл Справка

```
#include <cstdint>
#include <cmath>
```

Класове

class **PacMan::Point**

Именни пространства

namespace **PacMan**

C:/Users/Krusto/source/repos/PacManClone/PacManClone/Point.h

Вижте документацията за този файл.

```
1  #pragma once
2  #include <cstdint>
3  #include <cmath>
4  namespace PacMan {
5        class Point {
6        public:
10             Point() = default;
14             Point(float x, float y) : x(x), y(y) {}
18             Point(const Point&) = default;
19
20             Point& operator=(const Point& other) = default;
21             bool operator==(const Point& rhs) {
22                   return (this->x == rhs.x && this->y == rhs.y);
23             }
24             Point& operator+=(const Point& rhs)
25             {
26                   this->x += rhs.x;
27                   this->y += rhs.y;
28                   return *this;
29             }
30             Point& operator-=(const Point& rhs)
31             {
32                   this->x -= rhs.x;
33                   this->y -= rhs.y;
34                   return *this;
35             }
36             bool operator!=(const Point& rhs)
37             {
38                   return !(*this==rhs);
39             }
40             Point operator+(const Point& rhs)
41             {
42                   float x = this->x + rhs.x;
43                   float y = this->y + rhs.y;
44                   return { x,y };
45             }
46             Point& operator*(float rhs) {
47                   this->x *= rhs;
48                   this->y *= rhs;
49                   return *this;
50             }
51             Point GetTrunc() { return { trunc(floor(x)),trunc(floor(y)) }; };
52             float x{};
53             float y{};
54       };
55
56 }
```

C:/Users/Krusto/source/repos/PacManClone/PacManClone/WindowsFunctions.cpp Файл Справка

```
#include "WindowsFunctions.h"
```

Функции

void **GotoXY** (SHORT x, SHORT y)
POINT **GetCursorPosition** ()
void **ClearScreen** ()
void **ShowConsoleCursor** (bool showFlag)

Функции Документация

void ClearScreen ()

Функция която трие екрана

POINT GetCursorPosition ()

Функция която взима позицията на курсора

void GotoXY (SHORT *x*, SHORT *y*)

Функция която казва на курсора да отиде на x,y

void ShowConsoleCursor (bool *showFlag*)

Функцуя която показва или скрива курсора

C:/Users/Krusto/source/repos/PacManClone/PacManClone/WindowsFunctions.h Файл Справка

```
#include <Windows.h>
```

Функции

void **GotoXY** (SHORT x, SHORT y)
POINT **GetCursorPosition** ()
void **ClearScreen** ()
void **ShowConsoleCursor** (bool showFlag)

Функции Документация

void ClearScreen ()

Функция която трие екрана

POINT GetCursorPosition ()

Функция която взима позицията на курсора

void GotoXY (SHORT *x*, SHORT *y*)

Функция която казва на курсора да отиде на x,y

void ShowConsoleCursor (bool *showFlag*)

Функцуя която показва или скрива курсора

C:/Users/Krusto/source/repos/PacManClone/PacManClone/WindowsFunctions.h

Вижте документацията за този файл.1 #pragma once
2 #include <Windows.h>
3
7 void GotoXY(SHORT x, SHORT y);
11 POINT GetCursorPosition();
15 void ClearScreen();
19 void ShowConsoleCursor(bool showFlag);

Азбучен указател

INDEX