Pictorial Tetris

1.0.0

Wygenerowano przez Doxygen 1.8.11

Spis treści

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Rozdział 2

Indeks klas

2.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 3

Dokumentacja klas

3.1 Dokumentacja klasy Button

Reprezentuje przycisk wykorzystywany w menu gry.

```
#include <Button.h>
```

Diagram dziedziczenia dla Button

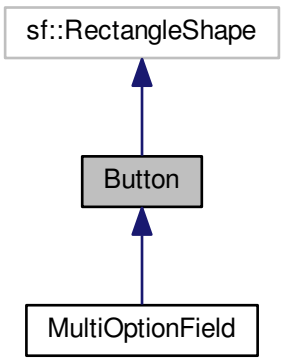
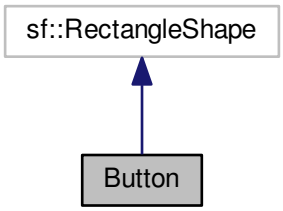


Diagram współpracy dla Button:



Metody publiczne

- Button (float width, float height, const sf::String &text="", unsigned int size=TEXT_SIZE)

  Konstruuje Button o podanych wymiarach i zawierający określony tekst.
- void draw (sf::RenderTarget &target, sf::RenderStates states) const override

  Renderuje przycisk.
- void select ()

  Ustawia przycisk jako zaznaczony w danym momencie.
- void unselect ()

  Ustawia przycisk jako nie zaznaczony w danym momencie.
- void setText (sf::Text &text)

  Ustawia obiekt Text widoczny na przycisku.
- void setString (const sf::String &string)

  Ustawia tekst w zawartym obiekcie klasy Text.
- void setPosition (float x, float y)

  Zmienia położenie przycisku na ekranie.

3.1.1 Opis szczegółowy

Reprezentuje przycisk wykorzystywany w menu gry.

3.1.2 Dokumentacja konstruktora i destruktora

3.1.2.1 Button::Button (float *width*, float *height*, const sf::String & *text* = " ", unsigned int *size* = TEXT_SIZE)

Konstruuje Button o podanych wymiarach i zawierający określony tekst.

Parametry

width	Szerokość przycisku w pikselach.
height	Wysokość przycisku w pikselach.
text	Tekst wyświetlany na przycisku.
size	Rozmiar tekstu.

3.1.3 Dokumentacja funkcji składowych

3.1.3.1 void Button::draw (sf::RenderTarget & *target,* sf::RenderStates *states*) const [override]

Renderuje przycisk.

Parametry

target	RenderTarget, do którego przycisk ma być renderowany.
states	Stany renderowania.

3.1.3.2 void Button::select ( )

Ustawia przycisk jako zaznaczony w danym momencie.

Zaznaczony przycisk ma inny kolor tła jak i tła tekstu.

3.1.3.3 void Button::setPosition (float *x,* float *y*)

Zmienia położenie przycisku na ekranie.

Parametry

x	Współrzędna na osi OX, w pikselach.
y	Współrzędna na osi OY, w pikselach.

3.1.3.4 void Button::setString (const sf::String & *string*)

Ustawia tekst w zawartym obiekcie klasy Text.

Rozmiar czcionki jest automatycznie dopasowywany do rozmiarów przycisku, jeżeli tekst by wystawał poza przycisk.

Parametry

string	Tekst do ustawienia.

3.1.3.5 void Button::setText (sf::Text & *text*)

Ustawia obiekt Text widoczny na przycisku.

Parametry

text	Text do ustawienia.

3.1.3.6 void Button::unselect ( )

Ustawia przycisk jako nie zaznaczony w danym momencie.

Odznaczony przycisk ma domyślny kolor tła jak i tła tekstu.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Button.h

3.2 Dokumentacja klasy Config

Reprezentuje konfigurację i ustawienia gry.

`#include <Config.h>`

Typy publiczne

- enum LEVEL { **EASY**, **NORMAL**, **HARD** }

Metody publiczne

- void setBlocks (unsigned int x)

  Ustawia ilość klocków na bok.
- unsigned int getBlocks () const

  Zwraca ilość klocków na bok.
- double getVelocity () const

  Zwraca szybkość opadania klocków.

Atrybuty publiczne

- LEVEL level = NORMAL

  Przechowuje aktualny poziom trudności gry.

Statyczne atrybuty publiczne

- static const int MIN_BLOCKS = 2

  Przechowuje minimalną możliwą ilość klocków na bok.
- static const int MAX_BLOCKS = 10

  Przechowuje maksymalną możliwą ilość klocków na bok.

3.2.1 Opis szczegółowy

Reprezentuje konfigurację i ustawienia gry.

3.2.2 Dokumentacja składowych wyliczanych

3.2.2.1 enum Config::LEVEL

Reprezentuje poziomy trudności gry.

3.2.3 Dokumentacja funkcji składowych

3.2.3.1 unsigned int Config::getBlocks ( ) const

Zwraca ilość klocków na bok.

Zwraca

Ilość klocków na bok.

3.2.3.2 double Config::getVelocity ( ) const

Zwraca szybkość opadania klocków.

Zwraca

Szybkość opadania klocków.

3.2.3.3 void Config::setBlocks (unsigned int *x*)

Ustawia ilość klocków na bok.

Parametry

x	Ilość klocków na bok.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Config.h

3.3 Dokumentacja klasy ConfigState

Reprezentuje stan bycia w konfiguracji gry.

```
#include <ConfigState.h>
```

Diagram dziedziczenia dla ConfigState

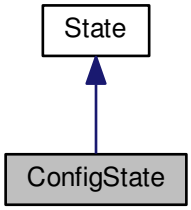
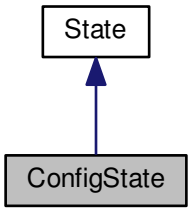


Diagram współpracy dla ConfigState:



Metody publiczne

- ConfigState ()

  Konstruuje ConfigState.
- void handleInput (sf::Event &event) override

  Przetwarza wejściowe zdarzenia pochodzące od użytkownika.
- void update (float dt) override

  Aktualizuje aktualny stan.
- void render () override

  Renderuje stan gry.

Dodatkowe Dziedziczone Składowe

3.3.1 Opis szczegółowy

Reprezentuje stan bycia w konfiguracji gry.

3.3.2 Dokumentacja funkcji składowych

3.3.2.1 void ConfigState::handleInput (sf::Event & *event*) [override], [virtual]

Przetwarza wejściowe zdarzenia pochodzące od użytkownika.

Parametry

event	Zdarzenie pochodzące od użytkownika.

Implementuje State.

3.3.2.2 void ConfigState::update (float *dt*) [override], [virtual]

Aktualizuje aktualny stan.

Parametry

dt	Czas renderowania i przetwarzania poprzedniej klatki gry.

Implementuje State.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- ConfigState.h

3.4 Dokumentacja klasy Game

Reprezentuje gre.

`#include <Game.h>`

Metody publiczne

- void run ()
  Uruchamia grę.

Statyczne metody publiczne

- static Game & getInstance ()
  Zwraca instancję obiektu Game.
- static View & getView ()
  Zwraca referencję do widoku View.
- static Config & getConfig ()
  Zwraca referencję do konfiguracji Config.
- static GameStateManager & getGSM ()
  Zwraca menadżer stanów GameStateManager.

Statyczne atrybuty publiczne

- static const std::string TITLE
  Przechowuje tytuł gry.

3.4.1 Opis szczegółowy

Reprezentuje gre.

Singleton – jednocześnie może występować tylko jedna instancja gry.

3.4.2 Dokumentacja funkcji składowych

3.4.2.1 static Config& Game::getConfig ( ) [static]

Zwraca referencję do konfiguracji Config.

Zwraca

Referencja do Config.

3.4.2.2 static GameStateManager& Game::getGSM ( ) [static]

Zwraca menadżer stanów GameStateManager.

Zwraca

Referencja do GameStateManager.

3.4.2.3 static Game& Game::getInstance ( ) [static]

Zwraca instancję obiektu Game.

Zwraca

Instancja obiektu Game.

3.4.2.4 static View& Game::getView ( ) [static]

Zwraca referencję do widoku View.

Zwraca

Referencja do View.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- Game.h

3.5 Dokumentacja klasy GameOverState

Reprezentuje stan końca gry.

`#include <GameOverState.h>`

Diagram dziedziczenia dla GameOverState

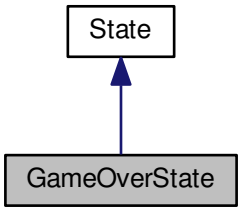
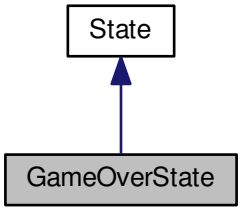


Diagram współpracy dla GameOverState:



Metody publiczne

- GameOverState ()

  Konstruuje GameOverState.
- void handleInput (sf::Event &event) override

  Przetwarza wejściowe zdarzenia pochodzące od użytkownika.
- void update (float dt) override

  Aktualizuje aktualny stan.
- void render () override

  Renderuje stan gry.

Dodatkowe Dziedziczone Składowe

3.5.1 Opis szczegółowy

Reprezentuje stan końca gry.

3.5.2 Dokumentacja funkcji składowych

3.5.2.1 void GameOverState::handleInput (sf::Event & *event*) [override], [virtual]

Przetwarza wejściowe zdarzenia pochodzące od użytkownika.

Parametry

event	Zdarzenie pochodzące od użytkownika.

Implementuje State.

3.5.2.2 void GameOverState::update (float *dt*) [override], [virtual]

Aktualizuje aktualny stan.

Parametry

dt	Czas renderowania i przetwarzania poprzedniej klatki gry.

Implementuje State.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- GameOverState.h

3.6 Dokumentacja klasy GameStateManager

Reprezentuje kontroler stanów aplikacji.

`#include <GameStateManager.h>`

Metody publiczne

- void handleInput (sf::Event &event)

  Przetwarza zdarzenie pochodzące od gracza przez aktualny stan aplikacji.
- void update (float dt)

  Aktualizuje aktualny stan aplikacji.

- void render ()

  Renderuje aktualn stan gry.
- void push (std::unique_ptr< State > state)

  Ustawia aktualny stan aplikacji.
- void pop ()

  Zamyka ostatni stan aplikacji.
- void set (std::unique_ptr< State > state)

  Zamyka poprzednie stany aplikacji i ustawia nowy.

3.6.1 Opis szczegółowy

Reprezentuje kontroler stanów aplikacji.

3.6.2 Dokumentacja funkcji składowych

3.6.2.1 void GameStateManager::handleInput (sf::Event & *event*)

Przetwarza zdarzenie pochodzące od gracza przez aktualny stan aplikacji.

Parametry

event	Zdarzenie pochodzące od gracza.

3.6.2.2 void GameStateManager::push (std::unique_ptr< State > *state*)

Ustawia aktualny stan aplikacji.

Parametry

state	Nowy stan do ustawienia.

3.6.2.3 void GameStateManager::set (std::unique_ptr< State > *state*)

Zamyka poprzednie stany aplikacji i ustawia nowy.

Parametry

state	Nowy stan do ustawienia.

3.6.2.4 void GameStateManager::update (float *dt*)

Aktualizuje aktualny stan aplikacji.

Parametry

dt	Czas renderowania i przetwarzania poprzedniej klatki gry.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- GameStateManager.h

3.7 Dokumentacja klasy HelpState

Reprezentuje stan bycia w pomocy.

`#include <HelpState.h>`

Diagram dziedziczenia dla HelpState

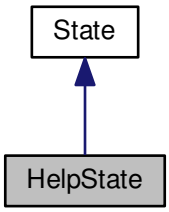
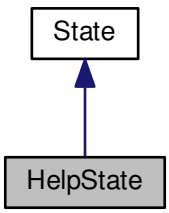


Diagram współpracy dla HelpState:



Metody publiczne

- HelpState ()=default

  Konstruuje HelpState.
- void handleInput (sf::Event &event) override

  Przetwarza wejściowe zdarzenia pochodzące od użytkownika.
- void update (float dt) override

  Aktualizuje aktualny stan.
- void render () override

  Renderuje stan gry.

Dodatkowe Dziedziczone Składowe

3.7.1 Opis szczegółowy

Reprezentuje stan bycia w pomocy.

3.7.2 Dokumentacja funkcji składowych

3.7.2.1 void HelpState::handleInput (sf::Event & *event*) [override], [virtual]

Przetwarza wejściowe zdarzenia pochodzące od użytkownika.

Parametry

event	Zdarzenie pochodzące od użytkownika.

Implementuje State.

3.7.2.2 void HelpState::update (float *dt*) [override], [virtual]

Aktualizuje aktualny stan.

Parametry

dt	Czas renderowania i przetwarzania poprzedniej klatki gry.

Implementuje State.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- HelpState.h

3.8 Dokumentacja klasy MenuState

Reprezentuje stan bycia w menu głównym.

`#include <MenuState.h>`

Diagram dziedziczenia dla MenuState

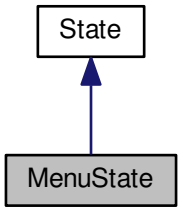
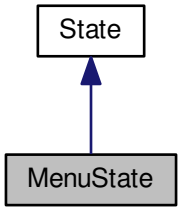


Diagram współpracy dla MenuState:



Metody publiczne

- MenuState ()

  Konstruuje MainMenu.
- void handleInput (sf::Event &event) override

  Przetwarza wejściowe zdarzenia pochodzące od użytkownika.
- void update (float dt) override

  Aktualizuje aktualny stan.
- void render () override

  Renderuje stan gry.

Dodatkowe Dziedziczone Składowe

3.8.1 Opis szczegółowy

Reprezentuje stan bycia w menu głównym.

3.8.2 Dokumentacja funkcji składowych

3.8.2.1 void MenuState::handleInput (sf::Event & *event*) [override], [virtual]

Przetwarza wejściowe zdarzenia pochodzące od użytkownika.

Parametry

event	Zdarzenie pochodzące od użytkownika.

Implementuje State.

3.8.2.2 void MenuState::update (float *dt*) [override], [virtual]

Aktualizuje aktualny stan.

Parametry

dt	Czas renderowania i przetwarzania poprzedniej klatki gry.

Implementuje State.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- MenuState.h

3.9 Dokumentacja klasy MultiOptionField

Reprezentuje przycisk z wyborem opcji.

```
#include <MultiOptionField.h>
```

Diagram dziedziczenia dla MultiOptionField

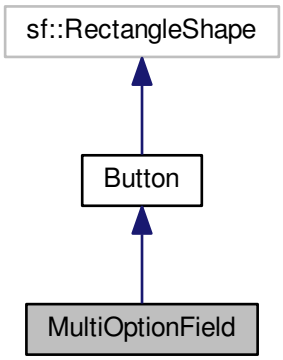
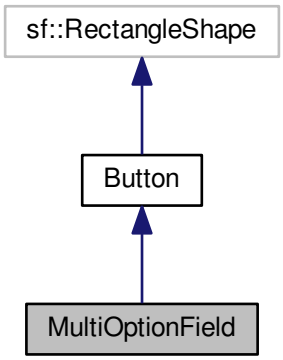


Diagram współpracy dla MultiOptionField:



Metody publiczne

- MultiOptionField (float width, float height, const sf::String &text="", unsigned int size=30, float percent=0.5)

  Konstruuje MultiOptionField o podanych rozmiarach i pozostałych parametrach.
- void draw (sf::RenderTarget &target, sf::RenderStates states) const

  Renderuje przycisk z wyborem opcji.
- void addOption (const sf::String &option)

  Dodaje nową opcję do wyboru.
- void setCurrent (unsigned int idx)

  Ustawia aktualną opcję.
- void setPosition (float x, float y)

Ustawia pozycję przycisku z wyborem opcji.

- void setOptionString (unsigned int idx, const sf::String &s)

  Ustawia tekst opcji o podanym indeksie.
- unsigned int selectNext ()

  Ustawia następną możliwą opcję.
- unsigned int selectPrevious ()

  Ustawia poprzednią możliwą opcję.

3.9.1 Opis szczegółowy

Reprezentuje przycisk z wyborem opcji.

3.9.2 Dokumentacja konstruktora i destruktora

3.9.2.1 MultiOptionField::MultiOptionField (float *width*, float *height*, const sf::String & *text* = " ", unsigned int *size* = 30, float *percent* = 0.5)

Konstruuje MultiOptionField o podanych rozmiarach i pozostałych parametrach.

Parametry

width	Szerokość w pikselach.
height	Wysokość w pikselach. Tekst widoczny na przycisku. Domyślnie pusty tekst. Rozmiar czcionki tekstu. Domyślnie 0.5. Procent zajmowanej szerokości. Domyślnie 0.5 (50%).

3.9.3 Dokumentacja funkcji składowych

3.9.3.1 void MultiOptionField::addOption (const sf::String & *option*)

Dodaje nową opcję do wyboru.

Jeżeli żadna opcjia nie istniała, dodaje ją i ustawia ją jako aktualną opcję.

Parametry

option	Opcja do dodania.

3.9.3.2 void MultiOptionField::draw (sf::RenderTarget & *target*, sf::RenderStates *states*) const

Renderuje przycisk z wyborem opcji.

Parametry

target	RenderTarged, do ktorego przycisk z wyborem opcji ma być renderowany.
states	Stany renderowania.

3.9.3.3 unsigned int MultiOptionField::selectNext ( )

Ustawia następną możliwą opcję.

Zwraca

Indeks nowej opcji.

3.9.3.4 unsigned int MultiOptionField::selectPrevious ( )

Ustawia poprzednią możliwą opcję.

Zwraca

Indeks nowej opcji.

3.9.3.5 void MultiOptionField::setCurrent (unsigned int *idx*)

Ustawia aktualną opcję.

Parametry

int	Indeks aktualnej opcji.

3.9.3.6 void MultiOptionField::setOptionString (unsigned int *idx*, const sf::String & *s*)

Ustawia tekst opcji o podanym indeksie.

Parametry

int	Indeks opcji.
s	Tekst do ustawienia.

3.9.3.7 void MultiOptionField::setPosition (float *x*, float *y*)

Ustawia pozycję przycisku z wyborem opcji.

Parametry

x	Współrzędna na osi OX w pikselach.
y	Współrzędna na osi OY w pikselach.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- MultiOptionField.h

3.10 Dokumentacja klasy PauseState

Reprezentuje stan pauzy.

`#include <PauseState.h>`

Diagram dziedziczenia dla PauseState

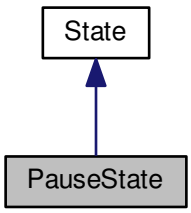
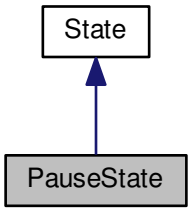


Diagram współpracy dla PauseState:



Metody publiczne

- PauseState ()=default

  Konstruuje PauseState.
- void handleInput (sf::Event &event) override

  Przetwarza wejściowe zdarzenia pochodzące od użytkownika.
- void update (float dt) override

  Aktualizuje aktualny stan.
- void render () override

  Renderuje stan gry.

Dodatkowe Dziedziczone Składowe

3.10.1 Opis szczegółowy

Reprezentuje stan pauzy.

3.10.2 Dokumentacja funkcji składowych

3.10.2.1 void PauseState::handleInput (sf::Event & *event*) [override], [virtual]

Przetwarza wejściowe zdarzenia pochodzące od użytkownika.

Parametry

event	Zdarzenie pochodzące od użytkownika.

Implementuje State.

3.10.2.2 void PauseState::update (float *dt*) [override], [virtual]

Aktualizuje aktualny stan.

Parametry

dt	Czas renderowania i przetwarzania poprzedniej klatki gry.

Implementuje State.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- PauseState.h

3.11 Dokumentacja klasy PlayState

Reprezentuje stan rozpoczęcia gry.

```
#include <PlayState.h>
```

Diagram dziedziczenia dla PlayState

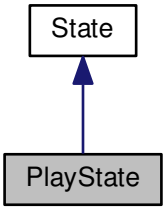
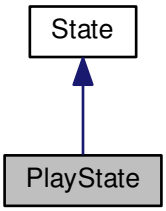


Diagram współpracy dla PlayState:



Metody publiczne

- PlayState ()

  Konstruuje PlayState.
- void handleInput (sf::Event &event) override

  Przetwarza wejściowe zdarzenia pochodzące od użytkownika.
- void update (float dt) override

  Aktualizuje aktualny stan.
- void render () override

  Renderuje stan gry.

Dodatkowe Dziedziczone Składowe

3.11.1 Opis szczegółowy

Reprezentuje stan rozpoczęcia gry.

3.11.2 Dokumentacja funkcji składowych

3.11.2.1 void PlayState::handleInput (sf::Event & *event*) [override], [virtual]

Przetwarza wejściowe zdarzenia pochodzące od użytkownika.

Parametry

event	Zdarzenie pochodzące od użytkownika.

Implementuje State.

3.11.2.2 void PlayState::update (float *dt*) [override], [virtual]

Aktualizuje aktualny stan.

Parametry

dt	Czas renderowania i przetwarzania poprzedniej klatki gry.

Implementuje State.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- PlayState.h

3.12 Dokumentacja klasy StartState

Diagram dziedziczenia dla StartState

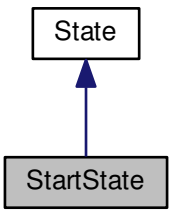
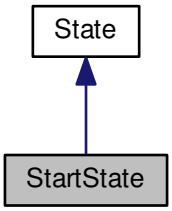


Diagram współpracy dla StartState:



Metody publiczne

- void handleInput (sf::Event &event) override

  Przetwarza wejściowe zdarzenia pochodzące od użytkownika.
- void update (float dt) override

  Aktualizuje aktualny stan.
- void render () override

  Renderuje stan gry.

Dodatkowe Dziedziczone Składowe

3.12.1 Dokumentacja funkcji składowych

3.12.1.1 void StartState::handleInput (sf::Event & *event*) [override], [virtual]

Przetwarza wejściowe zdarzenia pochodzące od użytkownika.

Parametry

event	Zdarzenie pochodzące od użytkownika.

Implementuje State.

3.12.1.2 void StartState::update (float *dt*) [override], [virtual]

Aktualizuje aktualny stan.

Parametry

dt	Czas renderowania i przetwarzania poprzedniej klatki gry.

Implementuje State.

Dokumentacja dla tej klasy została wygenerowana z pliku:

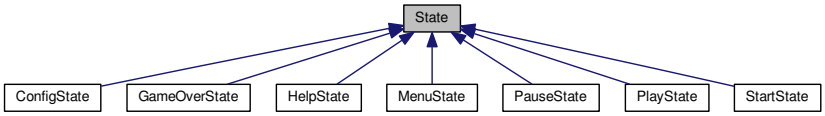
- StartState.h

3.13 Dokumentacja klasy State

Reprezentuje abstrakcyjny stan.

`#include <State.h>`

Diagram dziedziczenia dla State



Metody publiczne

- State ()=default

  Konstruuje obiekt abstrakcyjnego stanu. Konstruktor pusty.
- virtual ~State ()

  Wirtualny pusty destruktor.
- virtual void handleInput (sf::Event &event)=0

  Przetwarza wejściowe zdarzenia pochodzące od użytkownika. Czysto abstrakcyjna metoda.
- virtual void update (float dt)=0

  Aktualizuje aktualny stan. Czysto abstrakcyjna metoda.
- virtual void render ()=0

  Renderuje stan gry. Czysto abstrakcyjna metoda.

Atrybuty chronione

- bool updated {true}

3.13.1 Opis szczegółowy

Reprezentuje abstrakcyjny stan.

3.13.2 Dokumentacja funkcji składowych

3.13.2.1 virtual void State::handleInput (sf::Event & *event*) [pure virtual]

Przetwarza wejściowe zdarzenia pochodzące od użytkownika. Czysto abstrakcyjna metoda.

Parametry

event	Zdarzenie pochodzące od użytkownika.

Implementowany w PlayState, ConfigState, MenuState, HelpState, PauseState, GameOverState i StartState.

3.13.2.2 virtual void State::update (float *dt*) [pure virtual]

Aktualizuje aktualny stan. Czysto abstrakcyjna metoda.

Parametry

dt	Czas renderowania i przetwarzania poprzedniej klatki gry.

Implementowany w PlayState, ConfigState, MenuState, HelpState, PauseState, GameOverState i StartState.

3.13.3 Dokumentacja atrybutów składowych

3.13.3.1 bool State::updated {true} [protected]

Przechowuje informację o tym, czy stan został zaktualizowany.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- State.h

3.14 Dokumentacja klasy View

Reprezentuje widok.

`#include <View.h>`

Metody publiczne

- View ()
  Konstruuje View, tworząc okno gry, ustawiając rozdzielczość i inne niezbędne parametry.
- sf::RenderWindow & getWindow ()
  Zwraca okno gry.
- sf::Font & getFont ()
  Zwraca czcionkę wykorzystywaną w grze.
- sf::Texture & getTexture ()
  Zwraca teksturę zawierającą obrazek gry.
- const sf::Color & getBackgroundColor () const
  Zwraca kolor tła.
- void enableImg (unsigned int idx)

Ładuje obrazek do tablicy obrazków do podanego indeksu.

- void loadImg2Texture (unsigned int idx)

  Ustawia aktualny obrazek gry na podstawie indeksu tablicy z obrazkami.
- sf::Sprite resize (sf::Texture &texture, float width, float height)

  Przeskalowuje obrazek do odpowiednich wymiarów.
- unsigned int getHeight () const

  Zwraca wysokość okna gry.
- unsigned int getWidth () const

  Zwraca szerokość okna gry.

Atrybuty publiczne

- float time {0}

Statyczne atrybuty publiczne

- static const unsigned int IMAGES_COUNT = 4

3.14.1 Opis szczegółowy

Reprezentuje widok.

3.14.2 Dokumentacja funkcji składowych

3.14.2.1 void View::enableImg (unsigned int *idx*)

Ładuje obrazek do tablicy obrazków do podanego indeksu.

Tablica przyjmuję ściśle określoną maksymalną liczbę obrazków do pamięci. Dodatkowo każdy obrazek musi znajdować się w katalogu "data" i nazywać się "imgX.jpg", gdzie X jest także numerem indeksu w tablicy, gdzie zostanie zapisany.

Parametry

int	Indeks tablicy i końcówka nazwy pliku z obrazkiem.

3.14.2.2 const sf::Color& View::getBackgroundColor ( ) const

Zwraca kolor tła.

Zwraca

Kolor tła.

3.14.2.3 sf::Font& View::getFont ( )

Zwraca czcionkę wykorzystywaną w grze.

Zwraca

Czcionka.

3.14.2.4 unsigned int View::getHeight ( ) const

Zwraca wysokość okna gry.

Zwraca

Wysokość okna gry w pikselach.

3.14.2.5 sf::Texture& View::getTexture ( )

Zwraca teksturę zawierającą obrazek gry.

Zwraca

Tekstura zawierająca obrazek gry.

3.14.2.6 unsigned int View::getWidth ( ) const

Zwraca szerokość okna gry.

Zwraca

Szerokość okna gry w pikselach.

3.14.2.7 sf::RenderWindow& View::getWindow ( )

Zwraca okno gry.

Zwraca

Okno gry.

3.14.2.8 void View::loadImg2Texture (unsigned int *idx*)

Ustawia aktualny obrazek gry na podstawie indeksu tablicy z obrazkami.

Parametry

int	Indeks tablicy.

3.14.2.9 sf::Sprite View::resize (sf::Texture & *texture,* float *width,* float *height*)

Przeskalowuje obrazek do odpowiednich wymiarów.

Parametry

texture	Tekstura do przeskalowania.
width	Oczekiwana szerokość w pikselach.
height	Oczekiwana wysokość w pikselach.

Zwraca

Obiekt Sprite zawierający przeskalowany obrazek.

3.14.3 Dokumentacja atrybutów składowych

3.14.3.1 const unsigned int View::IMAGES_COUNT = 4 [static]

Przechowuje ilość przechowywanych obrazków w tablicy.

3.14.3.2 float View::time {0}

Przechowuje czas gry.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- View.h

Skorowidz