I Wanna Be The Scrub

Version 1.0

Generated by Doxygen 1.8.10

Mon Dec 21 2015 15:26:23

Contents

1 Prerequisites **1**

2 Hierarchical Index **3**

2.1 Class Hierarchy 3

3 Class Index **5**

3.1 Class List 5

4 Class Documentation **7**

4.1 ActiveBlock Class Reference 7

4.1.1 Constructor & Destructor Documentation 7

4.1.1.1 ActiveBlock(int x_p, int y_p, int w, int h, int amountOfFrames, std::string sprite↩ Sheet, SDL_Renderer ∗render) 7

4.2 Block Class Reference 8

4.2.1 Constructor & Destructor Documentation 8

4.2.1.1 Block(int x_p, int y_p, int w, int h, int amountOfFrames, std::string spriteSheet, SDL_Renderer ∗render) 8

4.3 Border Class Reference 9

4.4 Button Class Reference 9

4.4.1 Constructor & Destructor Documentation 10

4.4.1.1 Button(int x, int y, int w, int h, SDL_Renderer ∗render, std::string buttonImage, std::string newLevelName, int newStateIndex) 10

4.4.2 Member Function Documentation 10

4.4.2.1 checkForClick(int x, int y) 10

4.5 Checkpoint Class Reference 10

4.5.1 Constructor & Destructor Documentation 11

4.5.1.1 Checkpoint(int x_pos, int y_pos, SDL_Renderer ∗render) 11

4.6 Continuator Class Reference 11

4.6.1 Constructor & Destructor Documentation 11

4.6.1.1 Continuator(int x, int y, SDL_Renderer ∗render, std::string path) 11

4.7 DeathState Class Reference 12

4.7.1 Constructor & Destructor Documentation 12

4.7.1.1 DeathState(SDL_Window ∗win, SDL_Renderer ∗rend) 12

4.7.2 Member Function Documentation 13
4.7.2.1 cleanup() 13
4.7.2.2 handle(SDL_Event event, float deltaTime) 13
4.7.2.3 init() 13
4.7.2.4 update(float const &deltaTime) 13
4.8 Enemy Class Reference 13
4.8.1 Constructor & Destructor Documentation 14
4.8.1.1 Enemy(int x_p, int y_p, int w, int h, int amountOfFrames, std::string spriteSheet, SDL_Renderer ∗render, Player ∗player_ptr) 14
4.8.2 Member Function Documentation 14
4.8.2.1 moveLeft() 14
4.8.2.2 moveRight() 14
4.9 FallBlock Class Reference 15
4.9.1 Constructor & Destructor Documentation 15
4.9.1.1 FallBlock(int x_pos, int y_pos, SDL_Renderer ∗render, bool acti) 15
4.9.2 Member Function Documentation 15
4.9.2.1 activate() 15
4.9.2.2 update(float const &deltaTime) 15
4.10 FinishBlock Class Reference 16
4.11 GameObject Class Reference 16
4.11.1 Constructor & Destructor Documentation 17
4.11.1.1 GameObject(int x_p, int y_p, int w, int h, int amountOfFrames, std::string sprite↩Sheet, SDL_Renderer ∗render) 17
4.11.2 Member Function Documentation 18
4.11.2.1 intersect(GameObject ∗const &a, std::array< std::string, 4 > &result) 18
4.11.2.2 kill() 18
4.11.2.3 takeDmg() 18
4.11.2.4 updateGravity(float const &deltaTime) 18
4.11.2.5 updatePosition() 18
4.12 GameState Class Reference 18
4.13 GrassBlock Class Reference 19
4.13.1 Constructor & Destructor Documentation 20
4.13.1.1 GrassBlock(int x_pos, int y_pos, SDL_Renderer ∗render) 20
4.14 JumpBlock Class Reference 20
4.14.1 Constructor & Destructor Documentation 20
4.14.1.1 JumpBlock(int x_pos, int y_pos, SDL_Renderer ∗render) 20
4.15 MenuState Class Reference 21
4.15.1 Constructor & Destructor Documentation 21
4.15.1.1 MenuState(SDL_Window ∗win, SDL_Renderer ∗rend) 21
4.15.2 Member Function Documentation 22

4.15.2.1 cleanup() . . . 22
4.15.2.2 handle(SDL_Event event, float deltaTime) . . . 22
4.15.2.3 init() . . . 22
4.15.2.4 update(float const &deltaTime) . . . 22
4.16 MrRabbit Class Reference . . . 22
4.16.1 Constructor & Destructor Documentation . . . 23
4.16.1.1 MrRabbit(int x_pos, int y_pos, SDL_Renderer ∗render, Player ∗player_ptr) . . . 23
4.16.2 Member Function Documentation . . . 23
4.16.2.1 createObject(std::vector< GameObject ∗ > &map_objects) . . . 23
4.16.2.2 decideAction() . . . 23
4.16.2.3 update(float const &deltaTime) . . . 23
4.16.2.4 willCollide(std::vector< GameObject ∗ > const &objects) . . . 23
4.17 MrRabbitProjectile Class Reference . . . 24
4.18 Player Class Reference . . . 24
4.18.1 Constructor & Destructor Documentation . . . 25
4.18.1.1 Player(int x_pos, int y_pos, SDL_Renderer ∗render) . . . 25
4.18.2 Member Function Documentation . . . 25
4.18.2.1 createObject(std::vector< GameObject ∗ > &map_objects) . . . 25
4.18.2.2 handle(SDL_Event event, float deltaTime) . . . 25
4.18.2.3 shouldChangeSubLevel() . . . 25
4.18.2.4 update(float const &deltaTime) . . . 26
4.18.2.5 willCollide(std::vector< GameObject ∗ > const &objects) . . . 26
4.19 PlayerProjectile Class Reference . . . 26
4.19.1 Constructor & Destructor Documentation . . . 26
4.19.1.1 PlayerProjectile(int x, int y, int dir, SDL_Renderer ∗render, GameObject ∗creator_ptr) . . . 26
4.20 PlayState Class Reference . . . 27
4.20.1 Constructor & Destructor Documentation . . . 27
4.20.1.1 PlayState(SDL_Window ∗win, SDL_Renderer ∗rend) . . . 27
4.20.2 Member Function Documentation . . . 28
4.20.2.1 cleanup() . . . 28
4.20.2.2 createObject(std::string name, int x, int y, int extra, std::string path, SDL_Renderer ∗renderer) . . . 28
4.20.2.3 handle(SDL_Event event, float deltaTime) . . . 28
4.20.2.4 init() . . . 28
4.20.2.5 loadLevel(std::string level) . . . 28
4.20.2.6 update(float const &deltaTime) . . . 28
4.21 Projectile Class Reference . . . 29
4.21.1 Constructor & Destructor Documentation . . . 29
4.21.1.1 Projectile(int x_p, int y_p, int w, int h, int amountOfFrames, std::string spriteSheet, SDL_Renderer ∗render, int dir, GameObject ∗creator_ptr) . . . 29

4.21.2 Member Function Documentation 29
4.21.2.1 update(float const &deltaTime) 30
4.21.2.2 willCollide(std::vector< GameObject ∗ > const &objects) 30
4.22 Skeletron Class Reference 30
4.22.1 Constructor & Destructor Documentation 31
4.22.1.1 Skeletron(int x_pos, int y_pos, SDL_Renderer ∗render, Player ∗player_ptr, ActiveBlock ∗target_obj) 31
4.22.1.2 ∼Skeletron() 32
4.22.2 Member Function Documentation 32
4.22.2.1 createObject(std::vector< GameObject ∗ > &map_objects) 32
4.22.2.2 decideAction() 32
4.22.2.3 update(float const &deltaTime) 32
4.22.2.4 willCollide(std::vector< GameObject ∗ > const &objects) 32
4.23 SkeletronProjectile Class Reference 33
4.23.1 Constructor & Destructor Documentation 33
4.23.1.1 SkeletronProjectile(int x, int y, int dirX, int dirY, SDL_Renderer ∗render, Game←Object ∗creator_ptr) 33
4.24 SpeedDemon Class Reference 33
4.24.1 Constructor & Destructor Documentation 34
4.24.1.1 SpeedDemon(int x_pos, int y_pos, SDL_Renderer ∗render, Player ∗player_ptr) . 34
4.24.2 Member Function Documentation 34
4.24.2.1 decideAction() 34
4.24.2.2 willCollide(std::vector< GameObject ∗ > const &objects) 34
4.25 SpikeBlock Class Reference 34
4.25.1 Constructor & Destructor Documentation 35
4.25.1.1 SpikeBlock(int x_pos, int y_pos, SDL_Renderer ∗renderer, int acti) 35
4.25.2 Member Function Documentation 35
4.25.2.1 activate() 35
4.26 Sprite Class Reference 35
4.26.1 Constructor & Destructor Documentation 36
4.26.1.1 Sprite(int x_p, int y_p, int w, int h, int amountOfFrames, std::string spriteSheet, SDL_Renderer ∗render) 36
4.26.2 Member Function Documentation 36
4.26.2.1 draw(int const &direction) 36
4.26.2.2 loadSprite() 36
4.26.2.3 updateAnimation() 37
4.27 StateMachine Class Reference 37
4.27.1 Member Function Documentation 37
4.27.1.1 changeState(GameState ∗state) 37
4.27.1.2 cleanup() 37
4.27.1.3 init() 37

4.27.1.4 update() 37
4.28 TriggerBlock Class Reference 38
4.28.1 Constructor & Destructor Documentation 38
4.28.1.1 TriggerBlock(int x_pos, int y_pos, SDL_Renderer ∗renderer, ActiveBlock ∗target_obj) 38

Index **39**

Chapter 1

Prerequisites

- g++ compiler
- SDL2
- SDL2_image

Run instructions

Open terminal and navigate to game folder.

Run the following commands:

```
make
```

```
make run
```

This should open another window consisting of the game's menuscreen

Remove compiled game files

If for any reason you would like to removed the compiled game files:

```
make clean
```

In the game directory, will remove the game's compiled (.o) files.

Error handling

./game: No such file or directory

Check that the game compiled successfully, by typing `make` once more.

SDL functions not found

Check that you have a proper installation of SDL2 installed.

For further information regarding installation of SDL2, click `here`.

IMG_LOAD function not found

Check that you have a proper installation of SDL2_image installed.

For further information regarding installation of SDL2_image, click `here`.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- Button 9
- GameObject 16
  - Block 8
    - ActiveBlock 7
      - Continuator 11
      - FallBlock 15
      - FinishBlock 16
      - JumpBlock 20
      - SpikeBlock 34
      - TriggerBlock 38
    - GrassBlock 19
  - Border 9
  - Checkpoint 10
  - Enemy 13
    - MrRabbit 22
    - Skeletron 30
    - SpeedDemon 33
  - Player 24
  - Projectile 29
    - MrRabbitProjectile 24
    - PlayerProjectile 26
    - SkeletronProjectile 33
- GameState 18
  - DeathState 12
  - MenuState 21
  - PlayState 27
- Sprite 35
- StateMachine 37

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

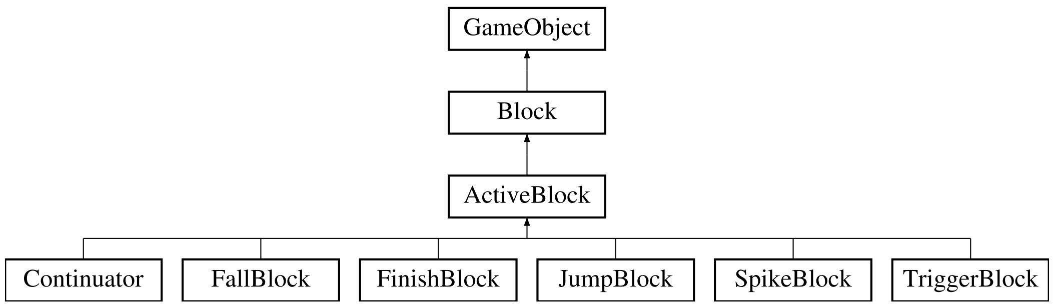
ActiveBlock 7
Block 8
Border 9
Button 9
Checkpoint 10
Continuator 11
DeathState 12
Enemy 13
FallBlock 15
FinishBlock 16
GameObject 16
GameState 18
GrassBlock 19
JumpBlock 20
MenuState 21
MrRabbit 22
MrRabbitProjectile 24
Player 24
PlayerProjectile 26
PlayState 27
Projectile 29
Skeletron 30
SkeletronProjectile 33
SpeedDemon 33
SpikeBlock 34
Sprite 35
StateMachine 37
TriggerBlock 38

Chapter 4

Class Documentation

4.1 ActiveBlock Class Reference

Inheritance diagram for ActiveBlock:



Public Member Functions

- ActiveBlock (int x_p, int y_p, int w, int h, int amountOfFrames, std::string spriteSheet, SDL_Renderer ∗render)

  Constructor for ActiveBlock.
- virtual void **activate** ()=0

Protected Attributes

- bool **active** {false}

Additional Inherited Members

4.1.1 Constructor & Destructor Documentation

4.1.1.1 ActiveBlock::ActiveBlock (int *x_p,* int *y_p,* int *w,* int *h,* int *amountOfFrames,* std::string *spriteSheet,* SDL_Renderer ∗ *render*)

Constructor for ActiveBlock.

Parameters

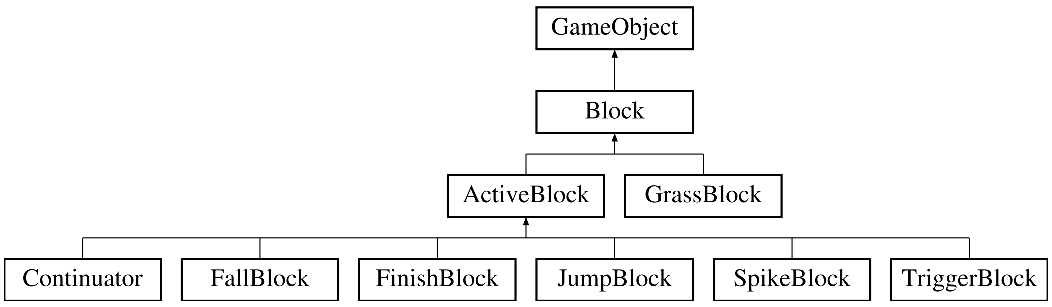
x_p	x position of block
y_p	y position of block
w	width of block
h	height of block
amountOf↩ Frames	amount of frames in blocks spritesheet
spriteSheet	location of spritesheet
render	renderer to draw to

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/ActiveBlock.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/ActiveBlock.cc

4.2 Block Class Reference

Inheritance diagram for Block:



Public Member Functions

- Block (int x_p, int y_p, int w, int h, int amountOfFrames, std::string spriteSheet, SDL_Renderer ∗render)

  Constructor for ActiveBlock.
- virtual void **update** (float const &deltaTime)
- void **willCollide** (std::vector< GameObject ∗ > const &objects)

Additional Inherited Members

4.2.1 Constructor & Destructor Documentation

4.2.1.1 Block::Block (int *x_p,* int *y_p,* int *w,* int *h,* int *amountOfFrames,* std::string *spriteSheet,* SDL_Renderer ∗ *render*)

Constructor for ActiveBlock.

Parameters

x_p	x position of block
y_p	y position of block

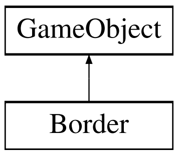
w	width of block
h	height of block
amountOf↩ Frames	amount of frames in blocks spritesheet
spriteSheet	location of spritesheet
render	renderer to draw to

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Block.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Block.cc

4.3 Border Class Reference

Inheritance diagram for Border:



Public Member Functions

- **Border** (int x_p, int y_p, int w, int h, SDL_Renderer ∗render)
- void **willCollide** (std::vector< GameObject ∗ > const &objects)
- void **update** (float const &deltaTime)

Additional Inherited Members

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Border.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Border.cc

4.4 Button Class Reference

Public Member Functions

- Button (int x, int y, int w, int h, SDL_Renderer ∗render, std::string buttonImage, std::string newLevelName, int newStateIndex)

  Constructor for Button class.
- bool checkForClick (int x, int y)

  Checks for click on button.

Public Attributes

- Sprite ∗ **sprite**
- int **clickStatechange**
- std::string **clickValue**

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Button::Button (int *x*, int *y*, int *w*, int *h*, SDL_Renderer ∗ *render*, std::string *buttonImage*, std::string *newLevelName*, int *newStateIndex*)

Constructor for Button class.

Parameters

x	x position of button
y	y position of button
w	width of button
h	height of button
render	renderer to draw to
buttonImage	location of image to draw to button
newLevelName	name of level to load on click
newStateIndex	index of state to change to

4.4.2 Member Function Documentation

4.4.2.1 bool Button::checkForClick (int *x*, int *y*)

Checks for click on button.

Compares x and y position of click with x and y position of button to see if the click occured on the button

Parameters

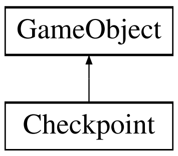
x	x position of click
y	y position of click

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Button.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Button.cc

4.5 Checkpoint Class Reference

Inheritance diagram for Checkpoint:



Public Member Functions

- Checkpoint (int x_pos, int y_pos, SDL_Renderer ∗render)

    Constructor for Checkpoint.
- void **update** (float const &deltaTime)
- void **willCollide** (std::vector< GameObject ∗ > const &objects)

Additional Inherited Members

4.5.1 Constructor & Destructor Documentation

4.5.1.1 Checkpoint::Checkpoint (int *x_pos*, int *y_pos*, SDL_Renderer ∗ *renderer*)

Constructor for Checkpoint.

Parameters

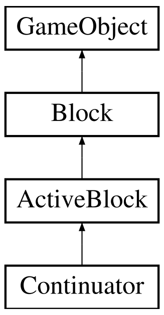
xPos	x position of checkpoint
yPos	y position of checkpoint renderer to draw to

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Checkpoint.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Checkpoint.cc

4.6 Continuator Class Reference

Inheritance diagram for Continuator:



Public Member Functions

- Continuator (int x, int y, SDL_Renderer ∗render, std::string path)

    Constructor for Continuator class.
- void **activate** ()
- std::string **getSubLevelPath** ()

Additional Inherited Members

4.6.1 Constructor & Destructor Documentation

4.6.1.1 Continuator::Continuator (int *x*, int *y*, SDL_Renderer ∗ *renderer*, std::string *path*)

Constructor for Continuator class.

Parameters

x	x position of continuator

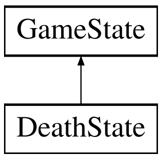
y	y position of continuator
renderer	renderer to draw to
path	location of new level to change to

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Continuator.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Continuator.cc

4.7 DeathState Class Reference

Inheritance diagram for DeathState:



Public Member Functions

- DeathState (SDL_Window ∗win, SDL_Renderer ∗rend)

    Constructor for DeathState.
- void init ()

    Initiator for Deathstate.
- void cleanup ()

    Cleans up pointers.
- void update (float const &deltaTime)

    Updator for DeathState.
- void handle (SDL_Event event, float deltaTime)

    Event handler for DeathState.

Public Attributes

- std::vector< Button ∗ > **menuItems**

Additional Inherited Members

4.7.1 Constructor & Destructor Documentation

4.7.1.1 DeathState::DeathState (SDL_Window ∗ *win,* SDL_Renderer ∗ *rend*)

Constructor for DeathState.

Parameters

win	window to draw to
rend	renderer to draw to

4.7.2 Member Function Documentation

4.7.2.1 void DeathState::cleanup ( ) [virtual]

Cleans up pointers.

Deletes the background pointer

Implements GameState.

4.7.2.2 void DeathState::handle (SDL_Event *event*, float *deltaTime*) [virtual]

Event handler for DeathState.

Switches to play state if spacebar is pressed

Parameters

event	event to handle

Implements GameState.

4.7.2.3 void DeathState::init ( ) [virtual]

Initiator for Deathstate.

Loads image to screen on death

Implements GameState.

4.7.2.4 void DeathState::update (float const & *deltaTime*) [virtual]

Updator for DeathState.

Clears screen, draws the background, draws the buttons and renders it all on update

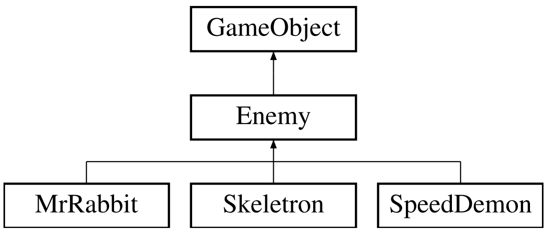
Implements GameState.

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/DeathState.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/DeathState.cc

4.8 Enemy Class Reference

Inheritance diagram for Enemy:



Public Member Functions

- Enemy (int x_p, int y_p, int w, int h, int amountOfFrames, std::string spriteSheet, SDL_Renderer ∗render, Player ∗player_ptr)

  Constructor for ActiveBlock.
- virtual void **decideAction** ()=0
- void moveLeft ()

  Moves to the left.
- void moveRight ()

  Moves to the right.

Protected Attributes

- Player ∗ **player**

Additional Inherited Members

4.8.1 Constructor & Destructor Documentation

4.8.1.1 Enemy::Enemy (int *x_p,* int *y_p,* int *w,* int *h,* int *amountOfFrames,* std::string *spriteSheet,* SDL_Renderer ∗ *render,* Player ∗ *player_ptr*)

Constructor for ActiveBlock.

Parameters

x_p	x position of block
y_p	y position of block
w	width of block
h	height of block
amountOf←↩ Frames	amount of frames in blocks spritesheet
spriteSheet	location of spritesheet
render	renderer to draw to
player_ptr	Pointer to the active player

4.8.2 Member Function Documentation

4.8.2.1 void Enemy::moveLeft ( )

Moves to the left.

Updates objects x position to move left on the game screen

4.8.2.2 void Enemy::moveRight ( )

Moves to the right.

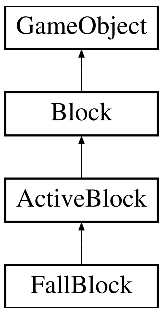
Updates objects x position to move right on the game screen

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Enemy.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Enemy.cc

4.9 FallBlock Class Reference

Inheritance diagram for FallBlock:



Public Member Functions

- FallBlock (int x_pos, int y_pos, SDL_Renderer ∗render, bool acti)

    Constructor for FallBlock.
- void activate ()

    Activator for FallBlock.
- void update (float const &deltaTime)

    Updator for FallBlock.

Additional Inherited Members

4.9.1 Constructor & Destructor Documentation

4.9.1.1 FallBlock::FallBlock (int *x_pos*, int *y_pos*, SDL_Renderer ∗ *render*, bool *acti*)

Constructor for FallBlock.

Parameters

xPos	x position of block
yPos	y position of block
render	renderer to draw to
acti	states if block is activatable by player

4.9.2 Member Function Documentation

4.9.2.1 void FallBlock::activate ( ) [virtual]

Activator for FallBlock.

Makes the block active and changes its type to the intended one

Implements ActiveBlock.

4.9.2.2 void FallBlock::update (float const & *deltaTime*) [virtual]

Updator for FallBlock.

Makes the block fall if it's active

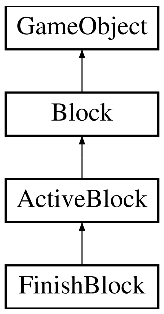
Reimplemented from Block.

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/FallBlock.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/FallBlock.cc

4.10 FinishBlock Class Reference

Inheritance diagram for FinishBlock:



Public Member Functions

- **FinishBlock** (int x_pos, int y_pos, SDL_Renderer ∗render, int visibleOnStart)
- void **activate** ()
- void **deActivate** ()

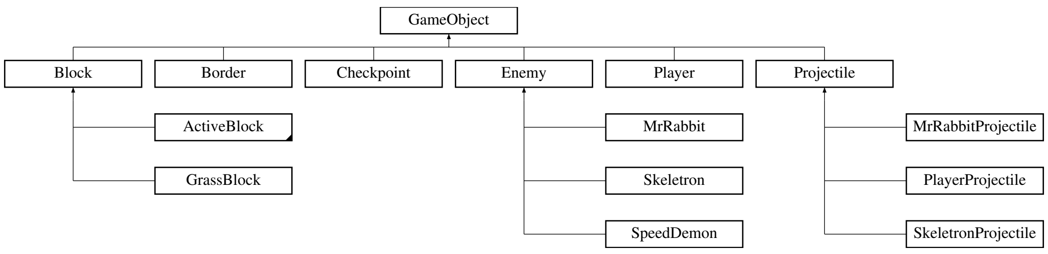
Additional Inherited Members

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/FinishBlock.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/FinishBlock.cc

4.11 GameObject Class Reference

Inheritance diagram for GameObject:



Public Member Functions

- GameObject (int x_p, int y_p, int w, int h, int amountOfFrames, std::string spriteSheet, SDL_Renderer ∗render)

    Constructor for GameObject.

- virtual void **handleCollision** (std::vector< std::pair< GameObject ∗, std::array< std::string, 4 > > > collidingObjects)=0
- virtual void **willCollide** (std::vector< GameObject ∗ > const &objects)=0
- virtual void **update** (float const &deltaTime)=0
- virtual void **createObject** (std::vector< GameObject ∗ > &map_objects)
- void updatePosition ()

  Updates position.
- void takeDmg ()

  Changes objects hp or kills it.
- void updateGravity (float const &deltaTime)

  Updates position based on gravity calculation.
- bool intersect (GameObject ∗const &a, std::array< std::string, 4 > &result)

  Checks for intersection and returns if a collision has occurred.
- void kill ()

  Kills the object.
- std::string **getType** () const
- double **getXVel** () const
- void **setXVel** (int const &new_xvel)
- double **getYVel** () const
- void **setYVel** (int const &new_yvel)
- bool **getDying** () const
- int **getDirection** () const
- int **getHealth** () const

Public Attributes

- int **xPos**
- int **yPos**
- bool **collidedThisUpdate** {false}
- Sprite ∗ **sprite**

Protected Attributes

- int **direction** {1}
- bool **dieNextUpdate** {false}
- double **yVel** {0}
- double **xVel** {0}
- int **health** {1}
- double **airTime** {0}
- std::string **type** {"GameObject"}

4.11.1 Constructor & Destructor Documentation

4.11.1.1 GameObject::GameObject (int *x_p*, int *y_p*, int *w*, int *h*, int *amountOfFrames*, std::string *spriteSheet*, SDL_Renderer ∗ *render*)

Constructor for GameObject.

Parameters

x_p	x position of block
y_p	y position of block
w	width of block
h	height of block
amountOf↩ Frames	amount of frames in blocks spritesheet
spriteSheet	location of spritesheet
render	renderer to draw to

4.11.2 Member Function Documentation

4.11.2.1 bool GameObject::intersect (GameObject ∗const & *a,* std::array< std::string, 4 > & *result*)

Checks for intersection and returns if a collision has occurred.

Checks for a collision between the object calling the function and another object. If a collision has occurred then the collision direction is calculated and saved for use later.

Returns

bool stating if collision has occured

4.11.2.2 void GameObject::kill ( )

Kills the object.

Kills the object by specifying that it should die on the next update

4.11.2.3 void GameObject::takeDmg ( )

Changes objects hp or kills it.

If the objects health is over one then it removes one hp from it. If the objects health is equal or less than one it kills it instead.

4.11.2.4 void GameObject::updateGravity (float const & *deltaTime*)

Updates position based on gravity calculation.

Simulates gravity by changing the objects y position

4.11.2.5 void GameObject::updatePosition ( )

Updates position.

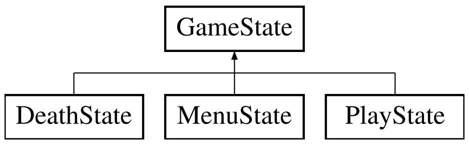
Updates the objects position by adding its velocity to its current x and y position values

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/GameObject.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/GameObject.cc

4.12 GameState Class Reference

Inheritance diagram for GameState:



Public Member Functions

- virtual void **init** ()=0
- virtual void **cleanup** ()=0
- virtual void **update** (float const &deltaTime)=0
- virtual void **handle** (SDL_Event event, float deltaTime)=0

Public Attributes

- std::string **level** {""}
- int **nextState** {}
- bool **pause** {false}
- int **playerX** {0}
- int **playerY** {0}

Protected Attributes

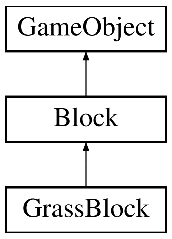
- Sprite $*$ **background**
- SDL_Window $*$ **window** = nullptr
- SDL_Renderer $*$ **renderer** = nullptr
- std::vector$<$ GameObject $*$ $>$ **objects**

The documentation for this class was generated from the following file:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/GameState.h

4.13 GrassBlock Class Reference

Inheritance diagram for GrassBlock:



Public Member Functions

- GrassBlock (int x_pos, int y_pos, SDL_Renderer $*$render)

  Constructor for GrassBlock.

Additional Inherited Members

4.13.1 Constructor & Destructor Documentation

4.13.1.1 GrassBlock::GrassBlock (int *x_pos*, int *y_pos*, SDL_Renderer ∗ *render*)

Constructor for GrassBlock.

Parameters

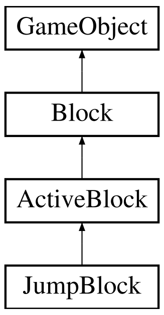
xPos	x position of block
yPos	y position of block
render	renderer to draw to

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/GrassBlock.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/GrassBlock.cc

4.14 JumpBlock Class Reference

Inheritance diagram for JumpBlock:



Public Member Functions

- JumpBlock (int x_pos, int y_pos, SDL_Renderer ∗render)

  Constructor for JumpBlock.
- void **activate** ()

Additional Inherited Members

4.14.1 Constructor & Destructor Documentation

4.14.1.1 JumpBlock::JumpBlock (int *x_pos*, int *y_pos*, SDL_Renderer ∗ *render*)

Constructor for JumpBlock.

Parameters

xPos	x position of block

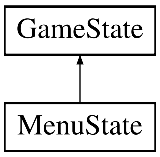
yPos	y position of block
render	renderer to draw to

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/JumpBlock.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/JumpBlock.cc

4.15 MenuState Class Reference

Inheritance diagram for MenuState:



Public Member Functions

- MenuState (SDL_Window ∗win, SDL_Renderer ∗rend)

  Constructor for MenuState.
- void init ()

  Initializer for MenuState.
- void cleanup ()

  Cleanup for MenuState.
- void update (float const &deltaTime)

  Updator for MenuState.
- void handle (SDL_Event event, float deltaTime)

  Event handler for MenuState.

Public Attributes

- std::vector< Button ∗ > **menuItems**

Additional Inherited Members

4.15.1 Constructor & Destructor Documentation

4.15.1.1 MenuState::MenuState (SDL_Window ∗ *win*, SDL_Renderer ∗ *rend*)

Constructor for MenuState.

Parameters

win	window to draw to
rend	renderer to draw to

4.15.2 Member Function Documentation

4.15.2.1 void MenuState::cleanup ( ) [virtual]

Cleanup for MenuState.

Deletes all buttons and the background.

Implements GameState.

4.15.2.2 void MenuState::handle (SDL_Event *event,* float *deltaTime*) [virtual]

Event handler for MenuState.

Handles mouse clicks and checks if the click occurred on a button and then loads the appropriate level.

Implements GameState.

4.15.2.3 void MenuState::init ( ) [virtual]

Initializer for MenuState.

Loads image to menu background and adds buttons for the different levels.

Implements GameState.

4.15.2.4 void MenuState::update (float const & *deltaTime*) [virtual]

Updator for MenuState.

Clears the screen and draws the background and buttons.

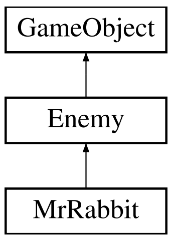
Implements GameState.

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/MenuState.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/MenuState.cc

4.16 MrRabbit Class Reference

Inheritance diagram for MrRabbit:



Public Member Functions

- MrRabbit (int x_pos, int y_pos, SDL_Renderer ∗render, Player ∗player_ptr)

  Constructor for MrRabbit.
- void update (float const &deltaTime)

Updator for MrRabbit.

- void decideAction ()

  Decides the next action for mr. rabbit.
- void createObject (std::vector< GameObject ∗ > &map_objects)

  Creates the mr. rabbit projectile.
- void willCollide (std::vector< GameObject ∗ > const &objects)

  Checks for collision with every object on the screen and handles it if collision has occurred.

Additional Inherited Members

4.16.1 Constructor & Destructor Documentation

4.16.1.1 MrRabbit::MrRabbit (int *x_pos*, int *y_pos*, SDL_Renderer ∗ *render*, Player ∗ *player_ptr*)

Constructor for MrRabbit.

Parameters

xPos	x position of mr. rabbit
yPos	y position of mr. rabbit
render	renderer to draw to
player_ptr	pointer to active player

4.16.2 Member Function Documentation

4.16.2.1 void MrRabbit::createObject (std::vector< GameObject ∗ > & *map_objects*) [virtual]

Creates the mr. rabbit projectile.

Creates mr. rabbits projectile if he is able to shoot and then resets the variable related to shooting

Reimplemented from GameObject.

4.16.2.2 void MrRabbit::decideAction ( ) [virtual]

Decides the next action for mr. rabbit.

Shoots next update and changes to correct direction if player is within a certain x or y distance. Moves right or left depending in the time since last movement.

Implements Enemy.

4.16.2.3 void MrRabbit::update (float const & *deltaTime*) [virtual]

Updator for MrRabbit.

Resets the bool that states if a collision has occurred this update. Decides the next action to take. Updates the time variable used to make decisions and the time since last shot variable. Updates x position of the object. Updates y position of the object according to gravity calculations.

Implements GameObject.

4.16.2.4 void MrRabbit::willCollide (std::vector< GameObject ∗ > const & *objects*) [virtual]

Checks for collision with every object on the screen and handles it if collision has occurred.

Creates a vector that will contain the object and the type of the object it collides with. Creates a vector with the resulting collision direction of the object it collides with. Loops through all objects on screen and checks for collision. Calls handleCollision() with the previously mentioned vector containing the object and its type.

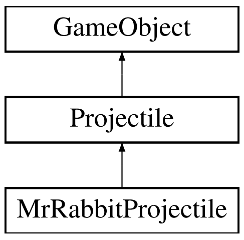
Implements GameObject.

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/MrRabbit.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/MrRabbit.cc

4.17 MrRabbitProjectile Class Reference

Inheritance diagram for MrRabbitProjectile:



Public Member Functions

- **MrRabbitProjectile** (int x, int y, int dir, SDL_Renderer ∗render, GameObject ∗creator_ptr)

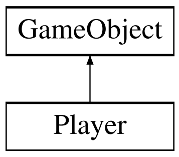
Additional Inherited Members

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/MrRabbitProjectile.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/MrRabbitProjectile.cc

4.18 Player Class Reference

Inheritance diagram for Player:



Public Member Functions

- Player (int x_pos, int y_pos, SDL_Renderer ∗render)

  Constructor for Player.
- void handle (SDL_Event event, float deltaTime)

  Event handler for Player.

- void willCollide (std::vector< GameObject * > const &objects)

  Checks for collision with every object on the screen and handles it if collision has occurred.
- void createObject (std::vector< GameObject * > &map_objects)

  Creates the players projectile.
- void update (float const &deltaTime)

  Updator for Player.
- std::string **getNextSubLevel** ()
- bool shouldChangeSubLevel ()

  Determines if it should (can) change sublevel.

Public Attributes

- int **jumpPower** {1}

Additional Inherited Members

4.18.1 Constructor & Destructor Documentation

4.18.1.1 Player::Player (int *x_pos*, int *y_pos*, SDL_Renderer * *render*)

Constructor for Player.

Parameters

xPos	x position of player
yPos	y position of player
render	renderer to draw to

4.18.2 Member Function Documentation

4.18.2.1 void Player::createObject (std::vector< **GameObject** * > & *map_objects*) [virtual]

Creates the players projectile.

Creates player projectile if he is able to shoot and then resets the variable related to shooting

Reimplemented from GameObject.

4.18.2.2 void Player::handle (SDL_Event *event*, float *deltaTime*)

Event handler for Player.

Resets x velocity to zero when movement keys are no longer pressed. Resets boolean that states if projectile can be shot next update when spacebar is no longer pressed. Calls appropriate move or jump functions if movement keys are presesed. Calls shoot function if space bar is pressed. Kills player if R is pressed.

4.18.2.3 bool Player::shouldChangeSubLevel ( )

Determines if it should (can) change sublevel.

Returns

if the next sublevel isn't an empty string it returns true.

4.18.2.4 void Player::update (float const & *deltaTime*) [virtual]

Updator for Player.

Resets the bool that states if a collision has occurred this update. Updates the time since last shot. Updates x position of the object. Updates y position of the object according to gravity calculations.

Implements GameObject.

4.18.2.5 void Player::willCollide (std::vector< GameObject ∗ > const & *objects*) [virtual]

Checks for collision with every object on the screen and handles it if collision has occurred.

Creates a vector that will contain the object and the type of the object it collides with. Creates a vector with the resulting collision direction of the object it collides with. Loops through all objects on screen and checks for collision. Calls handleCollision() with the previously mentioned vector containing the object and its type.

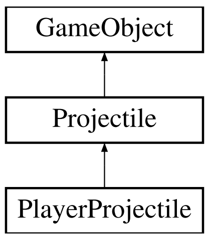
Implements GameObject.

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Player.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Player.cc

4.19 PlayerProjectile Class Reference

Inheritance diagram for PlayerProjectile:



Public Member Functions

- PlayerProjectile (int x, int y, int dir, SDL_Renderer ∗render, GameObject ∗creator_ptr)
  Constructor for PlayerProjectile.

Additional Inherited Members

4.19.1 Constructor & Destructor Documentation

4.19.1.1 PlayerProjectile::PlayerProjectile (int *x*, int *y*, int *dir*, SDL_Renderer ∗ *render*, GameObject ∗ *creator_ptr*)

Constructor for PlayerProjectile.

Parameters

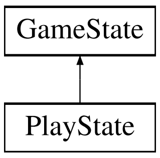
x	x position of projectile
y	y position of projectile
dir	direction the projectile spawns in
render	renderer to draw to
creator_ptr	pointer to the object that created the projectile

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/PlayerProjectile.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/PlayerProjectile.cc

4.20 PlayState Class Reference

Inheritance diagram for PlayState:



Public Member Functions

- PlayState (SDL_Window ∗win, SDL_Renderer ∗rend)

  Constructor for PlayState.
- void init ()

  Inititalizer for PlayState.
- void cleanup ()

  Cleanup for PlayState.
- void loadLevel (std::string level)

  Loads the objects from a .lvl file to the game field by calling createObject for every valid line.
- void createObject (std::string name, int x, int y, int extra, std::string path, SDL_Renderer ∗renderer)

  Creates an object on the game screen.
- void update (float const &deltaTime)

  Updater for PlayState.
- void handle (SDL_Event event, float deltaTime)

  Event handler for PlayState.

Additional Inherited Members

4.20.1 Constructor & Destructor Documentation

4.20.1.1 PlayState::PlayState (SDL_Window ∗ *win,* SDL_Renderer ∗ *rend*)

Constructor for PlayState.

Parameters

win	window to draw to
rend	renderer to draw to

4.20.2 Member Function Documentation

4.20.2.1 void PlayState::cleanup ( ) [virtual]

Cleanup for PlayState.

Deletes every object in the game objects vector and the background.

Implements GameState.

4.20.2.2 void PlayState::createObject (std::string *name*, int *x*, int *y*, int *extra*, std::string *path*, SDL_Renderer ∗ *renderer*)

Creates an object on the game screen.

Creates the appropriate object depending on the name specified and pushes it to the game object vector.

4.20.2.3 void PlayState::handle (SDL_Event *event*, float *deltaTime*) [virtual]

Event handler for PlayState.

Fetches a pointer to the player from the game objects vector. Moves to menu state if escape is pressed. Moves player to new sublevel if applicable. Sends events to player event handler.

Implements GameState.

4.20.2.4 void PlayState::init ( ) [virtual]

Inititalizer for PlayState.

Loads and draws the background image for the play state. Loads the chosen level.

Implements GameState.

4.20.2.5 void PlayState::loadLevel (std::string *level*)

Loads the objects from a .lvl file to the game field by calling createObject for every valid line.

Clears the map objects vector and reserves enough space to hold all objects. Outputs a status message to the console. Initializes variables needed to create objects. Opens the level file. Uses stringstreams to load each line into the preivously created variables. Sends the variables to createObject() if the object is valid. Creates borders around the game screen.

4.20.2.6 void PlayState::update (float const & *deltaTime*) [virtual]

Updater for PlayState.

Clears the screen. Draws the background. Loops through every object on the game screen. If an object is moving, check for collision with it and every other object. Create projectiles for object if applicable. Update object. Draw object in the correct direction. Move to deathstate if player dies. Move to menustate if player finishes game. Delete object.

Implements GameState.

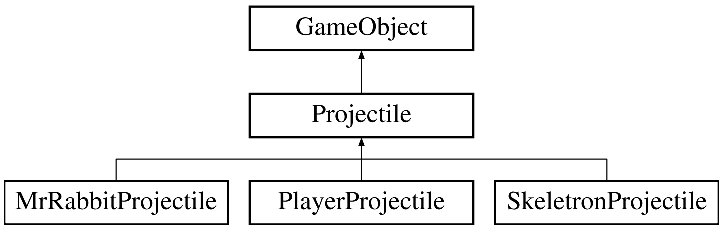
The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/PlayState.h

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/PlayState.cc

4.21 Projectile Class Reference

Inheritance diagram for Projectile:



Public Member Functions

- Projectile (int x_p, int y_p, int w, int h, int amountOfFrames, std::string spriteSheet, SDL_Renderer ∗render, int dir, GameObject ∗creator_ptr)

  Constructor for GameObject.
- virtual void **handleCollision** (std::vector< std::pair< GameObject ∗, std::array< std::string, 4 >>> collidingObjects)
- void update (float const &deltaTime)

  Updator for Projecile.
- void willCollide (std::vector< GameObject ∗ > const &objects)

  Checks for collision with every object on the screen and handles it if collision has occurred.

Additional Inherited Members

4.21.1 Constructor & Destructor Documentation

4.21.1.1 Projectile::Projectile (int *x_p,* int *y_p,* int *w,* int *h,* int *amountOfFrames,* std::string *spriteSheet,* SDL_Renderer ∗ *render,* int *dir,* GameObject ∗ *creator_ptr*)

Constructor for GameObject.

Parameters

x_p	x position of block
y_p	y position of block
w	width of block
h	height of block
amountOf↩ Frames	amount of frames in blocks spritesheet
spriteSheet	location of spritesheet
render	renderer to draw to
dir	direction to spawn the projectile in
creator_ptr	pointer to object that created projectile

4.21.2 Member Function Documentation

4.21.2.1 void Projectile::update (float const & *deltaTime*) [virtual]

Updator for Projecile.

Resets collision during this update bool. Updates position of projectile.

Implements GameObject.

4.21.2.2 void Projectile::willCollide (std::vector< GameObject ∗ > const & *objects*) [virtual]

Checks for collision with every object on the screen and handles it if collision has occurred.

Creates a vector that will contain the object and the type of the object it collides with. Creates a vector with the resulting collision direction of the object it collides with. Loops through all objects on screen and checks for collision. Calls handleCollision() with the previously mentioned vector containing the object and its type.

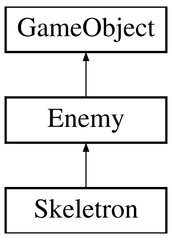
Implements GameObject.

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Projectile.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Projectile.cc

4.22 Skeletron Class Reference

Inheritance diagram for Skeletron:



Public Member Functions

- Skeletron (int x_pos, int y_pos, SDL_Renderer ∗render, Player ∗player_ptr, ActiveBlock ∗target_obj)

    Constructor for Skeletron.
- ∼Skeletron ()

    Deconstructor for Skeletron.
- void update (float const &deltaTime)

    Updator for Skeletron.
- void decideAction ()

    Decision making for Skeletron.
- void createObject (std::vector< GameObject ∗ > &map_objects)

    Projectile creation for Skeletron.
- void willCollide (std::vector< GameObject ∗ > const &objects)

    Checks for collision with every object on the screen and handles it if collision has occurred.

Additional Inherited Members

4.22.1 Constructor & Destructor Documentation

4.22.1.1 Skeletron::Skeletron (int *x_pos,* int *y_pos,* SDL_Renderer ∗ *render,* Player ∗ *player_ptr,* ActiveBlock ∗ *target_obj*)

Constructor for Skeletron.

Parameters

xPos	x position of mr. rabbit
yPos	y position of mr. rabbit
render	renderer to draw to
player_ptr	pointer to active player
target_obj	object to activate on death

4.22.1.2 Skeletron::∼Skeletron ( )

Deconstructor for Skeletron.

Deletes the pointer data members.

4.22.2 Member Function Documentation

4.22.2.1 void Skeletron::createObject (std::vector< **GameObject** ∗ > & *map_objects*) [virtual]

Projectile creation for Skeletron.

Creates skeletrons projectile if he is able to shoot and then resets the variable related to shooting.

Reimplemented from GameObject.

4.22.2.2 void Skeletron::decideAction ( ) [virtual]

Decision making for Skeletron.

If the player is nearby or a 1 in 200 chance occurs it will initialize a teleport. Changes direction depending on the players position. Determines where to shoot the projectile depending on the players y position.

Implements Enemy.

4.22.2.3 void Skeletron::update (float const & *deltaTime*) [virtual]

Updator for Skeletron.

Calculates and draws the health bar. Activates target object if dying. Draws the teleport indicator if a teleport is imminent and adds time to the time until teleport. Teleports if its time to teleport. Decides action. Updates x position. Updates y position.

Implements GameObject.

4.22.2.4 void Skeletron::willCollide (std::vector< **GameObject** ∗ > const & *objects*) [virtual]

Checks for collision with every object on the screen and handles it if collision has occurred.

Creates a vector that will contain the object and the type of the object it collides with. Creates a vector with the resulting collision direction of the object it collides with. Loops through all objects on screen and checks for collision. Calls handleCollision() with the previously mentioned vector containing the object and its type.

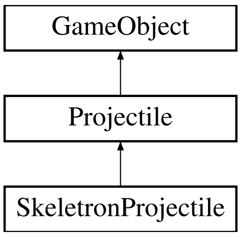
Implements GameObject.

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Skeletron.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Skeletron.cc

4.23 SkeletronProjectile Class Reference

Inheritance diagram for SkeletronProjectile:



Public Member Functions

- SkeletronProjectile (int x, int y, int dirX, int dirY, SDL_Renderer ∗render, GameObject ∗creator_ptr)

  Constructor for SkeletronProjectile.

Additional Inherited Members

4.23.1 Constructor & Destructor Documentation

4.23.1.1 SkeletronProjectile::SkeletronProjectile (int *x,* int *y,* int *dirX,* int *dirY,* SDL_Renderer ∗ *render,* GameObject ∗ *creator_ptr*)

Constructor for SkeletronProjectile.

Parameters

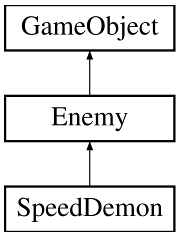
x	x position of projectile
y	y position of projectile
dirX	x direction to spawn projectile in
dirY	y direction to spawn projectile in
render	renderer to draw to
creator_ptr	pointer to object that created projectile

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/SkeletronProjectile.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/SkeletronProjectile.cc

4.24 SpeedDemon Class Reference

Inheritance diagram for SpeedDemon:



Public Member Functions

- SpeedDemon (int x_pos, int y_pos, SDL_Renderer ∗render, Player ∗player_ptr)

  Constructor for SpeedDemon.
- void update (float const &deltaTime)

  Update function for SpeedDemon.
- void willCollide (std::vector< GameObject ∗ > const &objects)

  Collision check for SpeedDemon.
- void decideAction ()

  Calculates next action.

Additional Inherited Members

4.24.1 Constructor & Destructor Documentation

4.24.1.1 SpeedDemon::SpeedDemon (int *x_pos,* int *y_pos,* SDL_Renderer ∗ *render,* Player ∗ *player_ptr*)

Constructor for SpeedDemon.

Parameters

xPos	x position of object
yPos	y position of object
player_ptr	pointer to extract its coordinates
render	renderer to draw to

4.24.2 Member Function Documentation

4.24.2.1 void SpeedDemon::decideAction ( ) [virtual]

Calculates next action.

If Player x is to the left of object and difference in y is less than 100, move to the left. If Player x is to the right of object and difference in y is less than 100, move to the right If Player is not within reach or same x, stand still.

Implements Enemy.

4.24.2.2 void SpeedDemon::willCollide (std::vector< GameObject ∗ > const & *objects*) [virtual]

Collision check for SpeedDemon.

Iterates the vector with objects in map and sends them to intersect

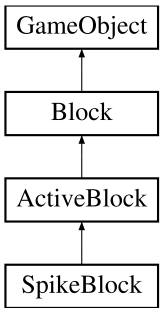
Implements GameObject.

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/SpeedDemon.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/SpeedDemon.cc

4.25 SpikeBlock Class Reference

Inheritance diagram for SpikeBlock:



Public Member Functions

- SpikeBlock (int x_pos, int y_pos, SDL_Renderer ∗renderer, int acti)

  Constructor for SpikeBlock.
- void activate ()

  Activate for SpikeBlock.
- void deActivate ()

  Hides the SpikeBlock.

Additional Inherited Members

4.25.1 Constructor & Destructor Documentation

4.25.1.1 SpikeBlock::SpikeBlock (int *x_pos*, int *y_pos*, SDL_Renderer ∗ *renderer*, int *acti*)

Constructor for SpikeBlock.

Parameters

xPos	x position of block
yPos	y position of block
render	renderer to draw to
active	If the object should be visible at start

4.25.2 Member Function Documentation

4.25.2.1 void SpikeBlock::activate ( ) [virtual]

Activate for SpikeBlock.

If the object at creation was set to be invisible, triggering via triggerblock will show it. If the object at creation was set to be visible, call deActivate function.

Implements ActiveBlock.

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/SpikeBlock.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/SpikeBlock.cc

4.26 Sprite Class Reference

Public Member Functions

- Sprite (int x_p, int y_p, int w, int h, int amountOfFrames, std::string spriteSheet, SDL_Renderer ∗render)

  Constructor for Sprite.
- void draw (int const &direction)

  Draws the current object.
- void updateAnimation ()

  Updates the animation.
- void loadSprite ()

  Loads the sprite from given path.
- void **setAnimationCurrent** (int const &newAnimationFrame)
- void **setAnimationProgress** (int const &newAnimationFrame)
- void **setAnimationStartIndex** (int const &newAnimationFrame)
- void **setAnimationEndIndex** (int const &newAnimationFrame)
- int **getAnimationCurrent** () const
- int **getAnimationProgress** () const
- int **getAnimationStartIndex** () const
- int **getAnimationEndIndex** () const
- SDL_Renderer ∗ **getRenderer** () const

Public Attributes

- SDL_Texture ∗ **texture**
- SDL_Rect **spriteClips** [8]
- SDL_Rect **spriteRect**

4.26.1 Constructor & Destructor Documentation

4.26.1.1 Sprite::Sprite (int *x_p*, int *y_p*, int *w*, int *h*, int *amountOfFrames*, std::string *spriteSheet*, SDL_Renderer ∗ *render*)

Constructor for Sprite.

Parameters

x_p	x position of object
y_p	y position of object
w	width of each animation frame
h	height of each animation frame
amountOf↩ Frames	amount of frames in blocks spritesheet
spriteSheet	location of spritesheet
render	renderer to draw to

4.26.2 Member Function Documentation

4.26.2.1 void Sprite::draw (int const & *direction*)

Draws the current object.

Depending on the objects direction, draws a flipped image

4.26.2.2 void Sprite::loadSprite ( )

Loads the sprite from given path.

Converts the image to and Texture∗ via Surface∗ Creates spriteclips depending on size and height of spriteRect

4.26.2.3 void Sprite::updateAnimation ( )

Updates the animation.

Depending on the current animation and how many animation it contains, it resets the animation value animation↩
Current is used to specifiy which parts of the spritesheet to display

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Sprite.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/Sprite.cc

4.27 StateMachine Class Reference

Public Member Functions

- void init ()

  Initializer for StateMachine.
- void cleanup ()

  Cleaner for StateMachine.
- void changeState (GameState ∗state)

  State changer StateMachine.
- GameState ∗ **getCurrentState** ()
- void update ()

  Calls current gamestates update function.

4.27.1 Member Function Documentation

4.27.1.1 void StateMachine::changeState (GameState ∗ *newState*)

State changer StateMachine.

Switches the member variable currentState to the gamestate pointer Initializes the new gamestate

4.27.1.2 void StateMachine::cleanup ( )

Cleaner for StateMachine.

Deallocates the different gamestates Deletes the Window Deletes the renderer

4.27.1.3 void StateMachine::init ( )

Initializer for StateMachine.

Creates windows with SCREEN_WIDTH and SCREEN_HEIGHT Creates a renderer to use for drawing Creates the different gamestates and adds them to a vector Changes the current state to display the menu Calls update for the current state

4.27.1.4 void StateMachine::update ( )

Calls current gamestates update function.

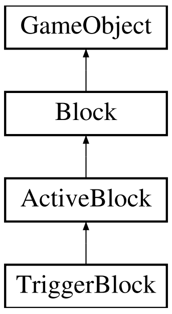
Calculates the time between frames Calls update function in the current gamestate with deltaTime as parameter

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/StateMachine.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/StateMachine.cc

4.28 TriggerBlock Class Reference

Inheritance diagram for TriggerBlock:



Public Member Functions

- TriggerBlock (int x_pos, int y_pos, SDL_Renderer ∗renderer, ActiveBlock ∗target_obj)

  Constructor for TriggerBlock.
- void activate ()

  Activator for TriggerBlock Calls the target objects activate function.

Additional Inherited Members

4.28.1 Constructor & Destructor Documentation

4.28.1.1 TriggerBlock::TriggerBlock (int *x_pos*, int *y_pos*, SDL_Renderer ∗ *renderer*, ActiveBlock ∗ *target_obj*)

Constructor for TriggerBlock.

Parameters

xPos	x position of block
yPos	y position of block
render	renderer to draw to
target_obj	Pointer to object which the trigger will activate.

The documentation for this class was generated from the following files:

- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/TriggerBlock.h
- /Users/lukas.vikstrom/Documents/TDP005/TDP005-Projekt/TriggerBlock.cc

Index

∼Skeletron
    Skeletron, 32

activate
    FallBlock, 15
    SpikeBlock, 35
ActiveBlock, 7
    ActiveBlock, 7

Block, 8
    Block, 8
Border, 9
Button, 9
    Button, 10
    checkForClick, 10

changeState
    StateMachine, 37
checkForClick
    Button, 10
Checkpoint, 10
    Checkpoint, 11
cleanup
    DeathState, 13
    MenuState, 22
    PlayState, 28
    StateMachine, 37
Continuator, 11
    Continuator, 11
createObject
    MrRabbit, 23
    PlayState, 28
    Player, 25
    Skeletron, 32

DeathState, 12
    cleanup, 13
    DeathState, 12
    handle, 13
    init, 13
    update, 13
decideAction
    MrRabbit, 23
    Skeletron, 32
    SpeedDemon, 34
draw
    Sprite, 36

Enemy, 13
    Enemy, 14
    moveLeft, 14
    moveRight, 14

FallBlock, 15
    activate, 15
    FallBlock, 15
    update, 15
FinishBlock, 16

GameObject, 16
    GameObject, 17
    intersect, 18
    kill, 18
    takeDmg, 18
    updateGravity, 18
    updatePosition, 18
GameState, 18
GrassBlock, 19
    GrassBlock, 20

handle
    DeathState, 13
    MenuState, 22
    PlayState, 28
    Player, 25

init
    DeathState, 13
    MenuState, 22
    PlayState, 28
    StateMachine, 37
intersect
    GameObject, 18

JumpBlock, 20
    JumpBlock, 20

kill
    GameObject, 18

loadLevel
    PlayState, 28
loadSprite
    Sprite, 36

MenuState, 21
    cleanup, 22
    handle, 22
    init, 22
    MenuState, 21
    update, 22
moveLeft

Enemy, 14
moveRight
Enemy, 14
MrRabbit, 22
createObject, 23
decideAction, 23
MrRabbit, 23
update, 23
willCollide, 23
MrRabbitProjectile, 24

PlayState, 27
cleanup, 28
createObject, 28
handle, 28
init, 28
loadLevel, 28
PlayState, 27
update, 28
Player, 24
createObject, 25
handle, 25
Player, 25
shouldChangeSubLevel, 25
update, 25
willCollide, 26
PlayerProjectile, 26
PlayerProjectile, 26
Projectile, 29
Projectile, 29
update, 29
willCollide, 30

shouldChangeSubLevel
Player, 25
Skeletron, 30
∼Skeletron, 32
createObject, 32
decideAction, 32
Skeletron, 31
update, 32
willCollide, 32
SkeletronProjectile, 33
SkeletronProjectile, 33
SpeedDemon, 33
decideAction, 34
SpeedDemon, 34
willCollide, 34
SpikeBlock, 34
activate, 35
SpikeBlock, 35
Sprite, 35
draw, 36
loadSprite, 36
Sprite, 36
updateAnimation, 36
StateMachine, 37
changeState, 37
cleanup, 37
init, 37
update, 37

takeDmg
GameObject, 18
TriggerBlock, 38
TriggerBlock, 38

update
DeathState, 13
FallBlock, 15
MenuState, 22
MrRabbit, 23
PlayState, 28
Player, 25
Projectile, 29
Skeletron, 32
StateMachine, 37
updateAnimation
Sprite, 36
updateGravity
GameObject, 18
updatePosition
GameObject, 18

willCollide
MrRabbit, 23
Player, 26
Projectile, 30
Skeletron, 32
SpeedDemon, 34