OOP Project

0.2

Generated by Doxygen 1.8.9.1

Tue Jan 20 2015 02:25:50

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

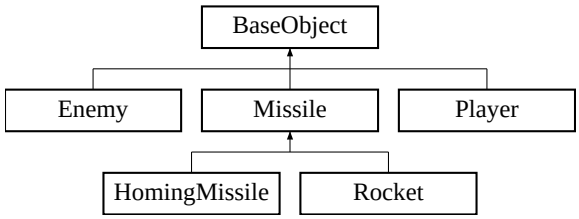
Class Documentation

3.1 BaseObject Class Reference

BaseObject class for the game. Every object in the game is inherited from this class.

`#include <BaseObject.h>`

Inheritance diagram for BaseObject:



Public Member Functions

- BaseObject ()
- BaseObject (const char ∗const _text, const ConsoleColor _fg, const ConsoleColor _bg, const short _x=0, const short _y=0)
- BaseObject (BaseObject const &_obj)
- BaseObject & operator= (BaseObject const &_obj)
- virtual ∼BaseObject ()
- void CalWH ()
- virtual void Input ()
- virtual void Update (int _frame)
- virtual void Render ()
- virtual bool Collides (const int _newX, const int _newY)
- bool OutOfBounds (const int _newX, const int _newY)

Accessors.

- const char ∗const **GetText** () const
- short **GetX** () const
- short **GetY** () const
- unsigned short **GetWidth** () const
- unsigned short **GetHeight** () const
- ConsoleColor **GetForeGround** () const
- ConsoleColor **GetBackGround** () const

- bool **GetAlive** () const
- OBJECT_ID **GetID** () const

Mutators.

- void **SetText** (const char *const _text)
- void **SetX** (const short _x)
- void **SetY** (const short _y)
- void **SetWidth** (const unsigned short _width)
- void **SetHeight** (const unsigned short _height)
- void **SetForeGround** (const ConsoleColor _color)
- void **SetBackGround** (const ConsoleColor _color)
- void **SetAlive** (bool _alive)
- void **SetID** (const OBJECT_ID _id)

3.1.1 Detailed Description

BaseObject class for the game. Every object in the game is inherited from this class.

Detailed description follows here.

Author

Junshu Chen

Date

Jan 2015

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BaseObject::BaseObject ( )

Default Constructor.

3.1.2.2 BaseObject::BaseObject (const char *const *_text*, const ConsoleColor *_fg*, const ConsoleColor *_bg*, const short *_x* = 0, const short *_y* = 0)

A Constructor sets up basic members.

Parameters

_text	an Image2D.
_fg	foreground color.
_bg	background color.
_x	x-position default is 0.
_y	y-position default is 0.

3.1.2.3 BaseObject::BaseObject (BaseObject const & *_obj*)

Copy Constructor.

3.1.2.4 BaseObject::~BaseObject ( ) [virtual]

virtual Destructor.

3.1.3 Member Function Documentation

3.1.3.1 void BaseObject::CalWH ( )

Calculate width and height of Image2D.

3.1.3.2 bool BaseObject::Collides (const int *_newX,* const int *_newY*) [virtual]

Take in a coordination to see if there is a collision or not. Different child class may have different colliding rule so virtual to allow override.

Parameters

_newX	x coordinate will be moved to in next frame.
_newY	y coordinate will be moved to in next frame.

Returns

a boolean indicate whether it will collide or not.

Reimplemented in Player, Enemy, and Missile.

3.1.3.3 void BaseObject::Input ( ) [virtual]

Handle User input.

Reimplemented in Player, Enemy, and Missile.

3.1.3.4 BaseObject & BaseObject::operator= (BaseObject const & *_obj*)

Overload Assignment Operator.

3.1.3.5 bool BaseObject::OutOfBounds (const int *_newX,* const int *_newY*)

Take in a coordination to see if it is out of bounds or not.

Parameters

_newX	x coordinate will be moved to in next frame.
_newY	y coordinate will be moved to in next frame.

Returns

a boolean indicate if it will be out of bounds.

3.1.3.6 void BaseObject::Render ( ) [virtual]

render current object.

Reimplemented in Player, Enemy, and Missile.

3.1.3.7 void BaseObject::Update (int *_frame*) [virtual]

Update current object.

Parameters

_frame	Global frame count.

Reimplemented in Player, Enemy, Missile, HomingMissile, and Rocket.

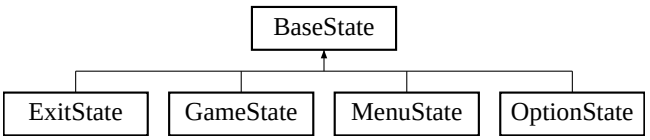
The documentation for this class was generated from the following files:

- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/BaseObject.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/BaseObject.cpp

3.2 BaseState Class Reference

`#include <BaseState.h>`

Inheritance diagram for BaseState:



Public Member Functions

- virtual void **Input** ()=0
- virtual void **Update** (int _frame)=0
- virtual void **Render** ()=0
- virtual void **Enter** ()=0
- virtual void **Exit** ()=0

Static Public Member Functions

- static GameInfo & GetInfo ()

3.2.1 Detailed Description

A abstract Base Class for states.

Author

Junshu Chen

Date

Jan 2015

3.2.2 Member Function Documentation

3.2.2.1 static GameInfo& BaseState::GetInfo ( ) [inline], [static]

Getinfo return Game setting info.

Returns

a reference to BaseState private member info.

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

- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/BaseState.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/BaseState.cpp

3.3 System::Console Class Reference

Static Public Member Functions

- static WORD **ForegroundColor** ()
- static void **ForegroundColor** (WORD attr)
- static WORD **BackgroundColor** ()
- static void **BackgroundColor** (WORD attr)
- static void **ResetColor** ()
- static int **WindowWidth** ()
- static int **WindowHeight** ()
- static void **SetWindowSize** (int columns, int rows)
- static void **SetBufferSize** (int columns, int rows)
- static int **CursorLeft** ()
- static int **CursorTop** ()
- static void **SetCursorPosition** (int left, int top)
- static void **Clear** ()
- static void **CursorVisible** (bool visible)
- static void **Lock** (bool lock)
- static void **EOLWrap** (bool on)
- static void **FlushKeys** ()
- static void **Show** (int x, int y, wchar_t symbol)
- static void **DrawBox** (int left, int top, int width, int height, bool dbl)
- static char const * **RandomName** ()
- static void **WordWrap** (int x, int y, int w, char const *const t)

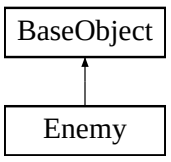
The documentation for this class was generated from the following files:

- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Console.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Console.cpp

3.4 Enemy Class Reference

`#include <Enemy.h>`

Inheritance diagram for Enemy:



Public Member Functions

- Enemy ()
- Enemy (const int _velocity, const char ∗const _text, const ConsoleColor _fg, const ConsoleColor _bg, const short _x=0, const short _y=0)
- ∼Enemy ()
- void Input ()
- void Update (int _frame)
- void Render ()
- bool Collides (const int _newX, const int _newY)

Accessors.

- int **GetVelocity** () const
- int **GetHP** () const

Mutators.

- void **SetVelocity** (const int _velocity)
- void **SetHP** (const int _hp)

3.4.1 Detailed Description

Enemy class to define enemy properties and behaviors.

Author

Junshu Chen

Date

Jan 2015

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Enemy::Enemy ( )

Default Constructor.

3.4.2.2 Enemy::Enemy (const int *_velocity,* const char ∗const *_text,* const ConsoleColor *_fg,* const ConsoleColor *_bg,* const short *_x* = 0, const short *_y* = 0)

A Constructor sets up every undefined members.

Parameters

_velocity.	
_score.	
_text	an Image2D.
_fg	foreground color.
_bg	background color.

_x	x-position default is 0.
_y	y-position default is 0.

3.4.2.3 Enemy::~Enemy ( )

Destructor.

3.4.3 Member Function Documentation

3.4.3.1 bool Enemy::Collides (const int *_newX*, const int *_newY*) [virtual]

Check collisions with player.

Reimplemented from BaseObject.

3.4.3.2 void Enemy::Input ( ) [virtual]

Handle enemy's input, so far it does nothing.

Reimplemented from BaseObject.

3.4.3.3 void Enemy::Render ( ) [virtual]

Render enemy on screen.

Reimplemented from BaseObject.

3.4.3.4 void Enemy::Update (int *_frame*) [virtual]

Update enemy movement and firing.

Parameters

_frame	global frame count.

Reimplemented from BaseObject.

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

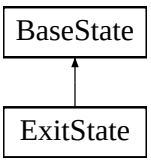
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Enemy.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Enemy.cpp

3.5 ExitState Class Reference

ExitState class exit state.

`#include <ExitState.h>`

Inheritance diagram for ExitState:



Public Member Functions

- void Enter ()
- void Render ()
- void **Update** (int _frame)
- void **Input** ()
- void **Exit** ()

Additional Inherited Members

3.5.1 Detailed Description

ExitState class exit state.

Author

Junshu Chen

Date

Jan 2015

3.5.2 Member Function Documentation

3.5.2.1 void ExitState::Enter ( ) [virtual]

Enter ExitState and set Game::play to false to terminate the game.

Implements BaseState.

3.5.2.2 void ExitState::Render ( ) [virtual]

Render Exit message on screen.

Implements BaseState.

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

- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/ExitState.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/ExitState.cpp

3.6 Game Class Reference

Game class.

`#include <Game.h>`

Public Member Functions

- Game ()
- ~Game ()
- void Play ()
- void Input ()
- void Update (int _frame)
- void Render ()

Static Public Member Functions

- static void ChangeState (STATE_TYPES _state)
- static void SetPlay (const bool _play)

3.6.1 Detailed Description

Game class.

Author

Junshu Chen

Date

Jan 2015

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Game::Game ( )

Default Constructor.

3.6.2.2 Game::~Game ( )

Default Destructor.

3.6.3 Member Function Documentation

3.6.3.1 void Game::ChangeState (STATE_TYPES *_state*) [static]

Change state based on [in]

Parameters

_state	the state will be switched to.

3.6.3.2 void Game::Input ( )

Call current state's Input function.

3.6.3.3 void Game::Play ( )

Main game loop calls Input, Update and Render functions.

3.6.3.4 void Game::Render ( )

Call current state's Render function.

3.6.3.5 static void Game::SetPlay (const bool *_play*) [inline], [static]

Toggle the Game::play by passing in boolean.

Parameters

_play	

3.6.3.6 void Game::Update (int *_frame*)

Call current state's Update function.

Parameters

_frame	global frame count.

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

- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Game.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Game.cpp

3.7 GameInfo Struct Reference

`#include <BaseState.h>`

Public Attributes

- int **diff**
- int **enemyNum**

3.7.1 Detailed Description

GameInfo struct to passing game setting between states.

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

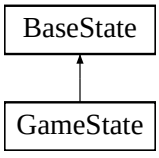
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/BaseState.h

3.8 GameState Class Reference

GameState class control actual gameplay.

`#include <GameState.h>`

Inheritance diagram for GameState:



Public Member Functions

- GameState ()
- ~GameState ()
- void Input ()

- void Update (int _frame)
- void Render ()
- void Enter ()
- void Exit ()
- void ReadObFromFile ()

Static Public Member Functions

- static vector< BaseObject * > * GetObjects ()

3.8.1 Detailed Description

GameState class control actual gameplay.

Author

Junshu Chen

Date

Jan 2015

3.8.2 Constructor & Destructor Documentation

3.8.2.1 GameState::GameState ( )

Constructor load player Image2D from file "Images.txt"

3.8.2.2 GameState::~GameState ( )

Destructor delete all game objects.

3.8.3 Member Function Documentation

3.8.3.1 void GameState::Enter ( ) [virtual]

Set up game based on the options player choose, the enemy number and difficulty.

Implements BaseState.

3.8.3.2 void GameState::Exit ( ) [virtual]

Save player's score in t scores.txt and scores.bin and return to main menu.

Implements BaseState.

3.8.3.3 static vector<BaseObject*>* GameState::GetObjects ( ) [inline], [static]

Return a point to gameObject array.

Returns

the address of the vector array of gameObjects

3.8.3.4 void GameState::Input ( ) [virtual]

call each game object's input function.

Implements BaseState.

3.8.3.5 void GameState::ReadObFromFile ( )

Load readInObjects Array from file "images.txt".

3.8.3.6 void GameState::Render ( ) [virtual]

call each game object's render function.

Implements BaseState.

3.8.3.7 void GameState::Update (int *_frame*) [virtual]

call each game object's update function.

Parameters

_frame	global frame count.

Implements BaseState.

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

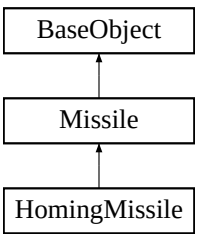
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/GameState.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/GameState.cpp

3.9 HomingMissile Class Reference

HomingMissile class derived from Missile Class.

`#include <HomingMissile.h>`

Inheritance diagram for HomingMissile:



Public Member Functions

- void Update (int _frame)

3.9.1 Detailed Description

HomingMissile class derived from Missile Class.

Author

Junshu Chen

Date

Jan 2015

3.9.2 Member Function Documentation

3.9.2.1 void HomingMissile::Update (int *_frame*) [virtual]

Updating velocity based on enemy position to achieve homing.

Parameters

_frame	global frame count.

Reimplemented from Missile.

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

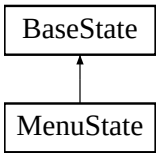
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/HomingMissile.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/HomingMissile.cpp

3.10 MenuState Class Reference

MenuState class control menu layout m.

`#include <MenuState.h>`

Inheritance diagram for MenuState:



Public Member Functions

- MenuState ()
- ~MenuState ()
- void Input ()
- void Update (int _frame)
- void Render ()
- void Enter ()
- void Exit ()

Additional Inherited Members

3.10.1 Detailed Description

MenuState class control menu layout m.

Author

Junshu Chen

Date

Jan 2015

3.10.2 Constructor & Destructor Documentation

3.10.2.1 MenuState::MenuState ( )

Constructor Layout Menu and read the menu art from MenuART.txt.

3.10.2.2 MenuState::~MenuState ( )

Destructor.

3.10.3 Member Function Documentation

3.10.3.1 void MenuState::Enter ( ) [virtual]

flush keys when entering menu state

Implements BaseState.

3.10.3.2 void MenuState::Exit ( ) [virtual]

flush keys when exiting menu state

Implements BaseState.

3.10.3.3 void MenuState::Input ( ) [virtual]

Handle user input enable user to select and enter different menu items.

Implements BaseState.

3.10.3.4 void MenuState::Render ( ) [virtual]

Render the menu

Implements BaseState.

3.10.3.5 void MenuState::Update (int *_frame*) [virtual]

Update keyboard buffer

Implements BaseState.

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

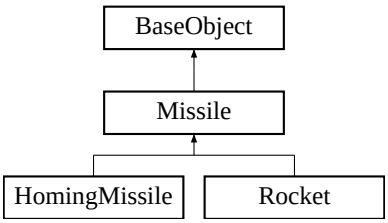
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/MenuState.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/MenuState.cpp

3.11 Missile Class Reference

Missile class for the game.

`#include <Missile.h>`

Inheritance diagram for Missile:



Public Member Functions

- Missile ()
- ~Missile ()
- void Input ()
- virtual void Update (int _frame)
- void Render ()
- bool Collides (const int _newX, const int _newY)

Accessors.

- void **SetVel** (int _x, int _y)
- void **SetXVel** (int _x)
- void **SetYVel** (int _y)

Mutators.

- int **GetXVel** () const
- int **GetYVel** () const
- void **GetVel** (int &_x, int &_y) const

3.11.1 Detailed Description

Missile class for the game.

Author

Junshu Chen

Date

Jan 2015

3.11.2 Constructor & Destructor Documentation

3.11.2.1 Missile::Missile ( )

Default Constructor.

3.11.2.2 Missile::~Missile ( )

Default Destructor.

3.11.3 Member Function Documentation

3.11.3.1 bool Missile::Collides (const int *_newX*, const int *_newY*) [virtual]

Check collision if it is a player missile collide with enemy and decrement enemy HP Add score to player. if it is an enemy missile collide with player and decrement player HP.

Reimplemented from BaseObject.

3.11.3.2 void Missile::Input ( ) [virtual]

Handle Missile's input which does nothing.

Reimplemented from BaseObject.

3.11.3.3 void Missile::Render ( ) [virtual]

Render missile on screen.

Reimplemented from BaseObject.

3.11.3.4 void Missile::Update (int *_frame*) [virtual]

update missile position based on its velocity virtual let child to override.

Reimplemented from BaseObject.

Reimplemented in HomingMissile, and Rocket.

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

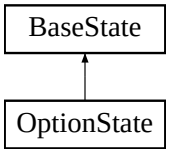
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Missile.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Missile.cpp

3.12 OptionState Class Reference

OptionState class.

`#include <OptionState.h>`

Inheritance diagram for OptionState:



Public Member Functions

- void Input ()
- void **Update** (int _frame)

- void Render ()
- void **Enter** ()
- void **Exit** ()

Additional Inherited Members

3.12.1 Detailed Description

OptionState class.

Author

Junshu Chen

Date

Jan 2015

3.12.2 Member Function Documentation

3.12.2.1 void OptionState::Input ( ) [virtual]

Enable Player to switch between menu items.

Implements BaseState.

3.12.2.2 void OptionState::Render ( ) [virtual]

Render option Menu.

Implements BaseState.

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

- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/OptionState.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/OptionState.cpp

3.13 System::PC Class Reference

Public Member Functions

- **PC** (bool go)
- void **Start** ()
- double **Finish** () const

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

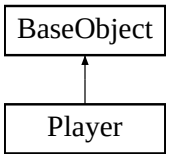
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Console.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Console.cpp

3.14 Player Class Reference

Player class for the game.

`#include <Player.h>`

Inheritance diagram for Player:



Public Member Functions

- Player ()
- Player (const char ∗const _name, int _score, int _diff, const char ∗const _text, const ConsoleColor _fg, const ConsoleColor _bg, const short _x=0, const short _y=0)
- Player (const Player &_obj)
- Player & operator= (const Player &_obj)
- ∼Player ()
- void Input ()
- void Update (int _frame)
- void Render ()
- bool Collides (const int _newX, const int _newY)

Accessors.

- const char ∗const **GetName** () const
- int **GetScore** () const
- int **GetDiff** () const
- int **GetNumofHM** () const
- int **GetNumofRK** () const
- bool **GetLaunched** () const
- int **GetHP** () const
- int **GetKillCount** () const

Mutators.

- void **SetName** (const char ∗const _name)
- void **SetScore** (const int _score)
- void **SetDiff** (const int _diff)
- void **SetNumofHM** (const int _numofHM)
- void **SetNumofRK** (const int _numofRK)
- void **SetLaunched** (const bool _launched)
- void **SetHP** (const int _hp)
- void **SetKillCount** (const int _killCount)

3.14.1 Detailed Description

Player class for the game.

Detailed description follows here.

Author

Junshu Chen

Date

Jan 2015

3.14.2 Constructor & Destructor Documentation

3.14.2.1 Player::Player ( )

Default Constructor.

3.14.2.2 Player::Player (const char *const _name, int _score, int _diff, const char *const _text, const ConsoleColor _fg, const ConsoleColor _bg, const short _x = 0, const short _y = 0)

A Constructor sets up every undefined members.

Parameters

_name	a C-string of name.
_score	score.
_diff	difficulty.
_text	an Image2D.
_fg	foreground color.
_bg	background color.
_x	x-position default is 0.
_y	y-position default is 0.

3.14.2.3 Player::Player (const Player & _obj)

Copy Constructor.

3.14.2.4 Player::~Player ( )

Destructor.

3.14.3 Member Function Documentation

3.14.3.1 bool Player::Collides (const int _newX, const int _newY) [virtual]

Check Player collision with enemies.

Parameters

_newX	x coordinate will be moved to in next frame.
_newY	y coordinate will be moved to in next frame.

Returns

a boolean indicate whether it will collide or not.

Reimplemented from BaseObject.

3.14.3.2 void Player::Input ( ) [virtual]

Handle player's input such as shooting missiles and movement.

Reimplemented from BaseObject.

3.14.3.3 Player & Player::operator= (const Player & *_obj*)

Overload Assignment Operator.

3.14.3.4 void Player::Render ( ) [virtual]

Render HUD and player's ship.

Reimplemented from BaseObject.

3.14.3.5 void Player::Update (int *_frame*) [virtual]

Update player status based on frames.

Parameters

_frame	global frame count.

Reimplemented from BaseObject.

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

- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Player.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Player.cpp

3.15 PlayerInfo Struct Reference

Player info struct for passing info between states and saving scores to file.

`#include <Player.h>`

Public Attributes

- char **buffer** [32]
- int **score**
- int **diff**
- int **killCount**

3.15.1 Detailed Description

Player info struct for passing info between states and saving scores to file.

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

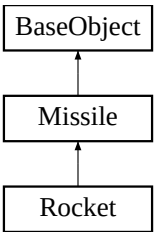
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Player.h

3.16 Rocket Class Reference

Rocket class inherited from Missile class.

`#include <Rocket.h>`

Inheritance diagram for Rocket:



Public Member Functions

- void Update (int _frame)

3.16.1 Detailed Description

Rocket class inherited from Missile class.

Author

Junshu Chen

Date

Jan 2015

3.16.2 Member Function Documentation

3.16.2.1 void Rocket::Update (int *_frame*) [virtual]

update missile position based on its velocity virtual let child to override.

Reimplemented from Missile.

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

- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Rocket.h
- C:/Users/Tom/OneDrive/Full Sail/Lectures/Object-Oriented Programming/OOP/OOP/Rocket.cpp

Index