

Doodle Jump

Generated by Doxygen 1.9.2

1 Advanced-Programming-DoodleJump	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Namespace Documentation	11
6.1 Model Namespace Reference	11
6.1.1 Detailed Description	11
6.1.2 Enumeration Type Documentation	12
6.1.2.1 Type	12
6.2 Observer Namespace Reference	12
6.2.1 Detailed Description	12
6.3 Settings Namespace Reference	12
6.3.1 Detailed Description	12
6.3.2 Enumeration Type Documentation	13
6.3.2.1 Difficulty	13
6.4 Utils Namespace Reference	13
6.4.1 Detailed Description	13
7 Class Documentation	15
7.1 Model::AbstractFactory Class Reference	15
7.2 Model::Background Class Reference	15
7.2.1 Member Function Documentation	16
7.2.1.1 getType()	16
7.2.1.2 move()	16
7.3 View::BackgroundView Class Reference	16
7.3.1 Member Function Documentation	17
7.3.1.1 handleEvent() [1/2]	17
7.3.1.2 handleEvent() [2/2]	17
7.4 Model::Bonus Class Reference	17
7.4.1 Member Function Documentation	18
7.4.1.1 getMSort()	18
7.4.1.2 getType()	18
7.4.1.3 setMSort()	18
7.5 Controller::BonusController Class Reference	19

7.6 View::BonusView Class Reference	19
7.7 Utils::Camera Class Reference	20
7.7.1 Detailed Description	21
7.7.2 Member Function Documentation	21
7.7.2.1 getInstance()	21
7.7.2.2 getMaxHeight()	21
7.7.2.3 getWindowDimensions()	22
7.7.2.4 getWorldDimensions()	22
7.7.2.5 getX()	22
7.7.2.6 getY()	22
7.7.2.7 inverseTransform()	22
7.7.2.8 isMaxHeight()	23
7.7.2.9 move()	23
7.7.2.10 operator=()	23
7.7.2.11 setWindowDimensions()	24
7.7.2.12 setWorldDimensions()	24
7.7.2.13 transform()	24
7.8 Model::CollisionBox Class Reference	25
7.9 CollisionEvent Class Reference	25
7.9.1 Member Function Documentation	26
7.9.1.1 send()	26
7.10 View::ConcreteFactory Class Reference	26
7.10.1 Member Function Documentation	27
7.10.1.1 createBackground()	27
7.10.1.2 createHorizontalPlatform()	27
7.10.1.3 createJetpack()	27
7.10.1.4 createPlayer()	27
7.10.1.5 createScore()	27
7.10.1.6 createSpring()	28
7.10.1.7 createStaticPlatform()	28
7.10.1.8 createTemporaryPlatform()	28
7.10.1.9 createVerticalPlatform()	28
7.11 DrawEvent Class Reference	28
7.11.1 Member Function Documentation	29
7.11.1.1 send()	29
7.12 Model::Entity Class Reference	29
7.12.1 Detailed Description	31
7.12.2 Member Function Documentation	31
7.12.2.1 getHeight()	31
7.12.2.2 getType()	31
7.12.2.3 getX()	31
7.12.2.4 getY()	32

7.12.2.5 move() [1/2]	32
7.12.2.6 move() [2/2]	32
7.12.2.7 setHeight()	32
7.12.2.8 setWidth()	33
7.12.2.9 setX()	33
7.12.2.10 setY()	33
7.12.2.11 visit()	33
7.12.3 Member Data Documentation	34
7.12.3.1 mHeight	34
7.12.3.2 mWidth	34
7.12.3.3 mX	34
7.12.3.4 mY	34
7.13 Event Class Reference	35
7.14 Utils::Exception Class Reference	35
7.15 Utils::FileException Class Reference	36
7.16 Game Class Reference	36
7.16.1 Detailed Description	37
7.17 HighScore Class Reference	37
7.18 HighScoreScore Struct Reference	37
7.19 Model::HorizontalPlatform Class Reference	38
7.19.1 Member Function Documentation	38
7.19.1.1 getType()	39
7.19.1.2 move()	39
7.20 Controller::IController Class Reference	39
7.20.1 Member Function Documentation	40
7.20.1.1 handleEvent() [1/3]	40
7.20.1.2 handleEvent() [2/3]	40
7.20.1.3 handleEvent() [3/3]	40
7.20.1.4 onTrigger()	40
7.21 IEventHandler Class Reference	41
7.22 View::IView Class Reference	41
7.22.1 Member Function Documentation	42
7.22.1.1 handleEvent() [1/4]	42
7.22.1.2 handleEvent() [2/4]	42
7.22.1.3 handleEvent() [3/4]	42
7.22.1.4 handleEvent() [4/4]	42
7.22.1.5 onTrigger()	43
7.23 Visitor::IVisitor Class Reference	43
7.24 Model::Jetpack Class Reference	44
7.24.1 Member Function Documentation	44
7.24.1.1 getType()	44
7.24.1.2 isBonus()	45

7.24.1.3 isRemovable()	45
7.24.1.4 move()	45
7.24.1.5 visit()	45
7.25 KeyPressedEvent Class Reference	45
7.25.1 Member Function Documentation	46
7.25.1.1 send()	46
7.26 MoveEvent Class Reference	46
7.26.1 Member Function Documentation	46
7.26.1.1 send()	47
7.27 NewDifficultyEvent Class Reference	47
7.27.1 Member Function Documentation	47
7.27.1.1 send()	47
7.28 NewMaxHeightEvent Class Reference	48
7.28.1 Member Function Documentation	48
7.28.1.1 send()	48
7.29 Observer::Observer Class Reference	48
7.29.1 Detailed Description	49
7.29.2 Member Function Documentation	49
7.29.2.1 onTrigger()	49
7.30 OutOfViewEvent Class Reference	49
7.30.1 Member Function Documentation	50
7.30.1.1 send()	50
7.31 Model::Platform Class Reference	50
7.31.1 Member Function Documentation	50
7.31.1.1 getType()	51
7.31.1.2 move()	51
7.32 Controller::PlatformController Class Reference	51
7.33 View::PlatformView Class Reference	52
7.33.1 Member Function Documentation	52
7.33.1.1 handleEvent()	52
7.34 Model::Player Class Reference	53
7.34.1 Detailed Description	53
7.34.2 Member Function Documentation	54
7.34.2.1 accept()	54
7.34.2.2 getType()	54
7.34.2.3 handleEvent()	54
7.34.2.4 move()	54
7.34.2.5 onTrigger()	55
7.35 Controller::PlayerController Class Reference	55
7.35.1 Member Function Documentation	55
7.35.1.1 handleEvent() [1/3]	55
7.35.1.2 handleEvent() [2/3]	56

7.35.1.3 handleEvent() [3/3]	56
7.36 View::PlayerView Class Reference	56
7.37 Utils::Random Class Reference	57
7.37.1 Detailed Description	57
7.37.2 Member Function Documentation	57
7.37.2.1 getInstance()	57
7.37.2.2 operator=()	58
7.37.2.3 random()	58
7.38 Utils::Resourceholder< Type > Struct Template Reference	58
7.39 Utils::Resourcemanager Class Reference	59
7.40 Model::Score Class Reference	59
7.40.1 Member Function Documentation	60
7.40.1.1 getScore()	60
7.40.1.2 getType()	60
7.40.1.3 handleEvent() [1/2]	60
7.40.1.4 handleEvent() [2/2]	60
7.40.1.5 move()	61
7.40.1.6 onTrigger()	61
7.41 View::ScoreView Class Reference	61
7.42 Model::Spring Class Reference	62
7.42.1 Member Function Documentation	62
7.42.1.1 getType()	62
7.42.1.2 isBonus()	62
7.42.1.3 isRemovable()	63
7.42.1.4 move()	63
7.42.1.5 visit()	63
7.43 Model::StaticPlatform Class Reference	63
7.43.1 Member Function Documentation	64
7.43.1.1 getType()	64
7.43.1.2 move()	64
7.44 StopBonusEvent Class Reference	64
7.44.1 Member Function Documentation	65
7.44.1.1 send()	65
7.45 Utils::Stopwatch Class Reference	65
7.45.1 Detailed Description	66
7.45.2 Member Function Documentation	66
7.45.2.1 getDelta()	66
7.45.2.2 getInstance()	66
7.45.2.3 lap()	67
7.45.2.4 operator=()	67
7.46 Observer::Subject Class Reference	67
7.46.1 Detailed Description	68

7.46.2 Member Function Documentation	68
7.46.2.1 add()	69
7.47 Model::TemporaryPlatform Class Reference	69
7.47.1 Member Function Documentation	69
7.47.1.1 getType()	69
7.47.1.2 move()	70
7.48 Utils::Utilities Class Reference	70
7.48.1 Detailed Description	70
7.48.2 Member Function Documentation	70
7.48.2.1 checkCollision()	70
7.48.2.2 checkWeight()	71
7.49 Model::VerticalPlatform Class Reference	71
7.49.1 Member Function Documentation	72
7.49.1.1 getType()	72
7.49.1.2 move()	72
7.50 World Class Reference	72
7.50.1 Detailed Description	73
7.50.2 Member Function Documentation	73
7.50.2.1 addEntity()	73
7.50.2.2 events()	74
7.50.2.3 spawnBonus()	74
7.50.2.4 spawnEntity()	74
7.50.2.5 spawnPlatform()	75
8 File Documentation	77
8.1 AbstractFactory.h	77
8.2 BonusController.h	77
8.3 IController.h	78
8.4 PlatformController.h	78
8.5 PlayerController.h	79
8.6 Event.h	79
8.7 HighScore.h	82
8.8 IVisitor.h	83
8.9 Background.h	83
8.10 Bonus.h	83
8.11 Entity.h	84
8.12 HorizontalPlatform.h	85
8.13 Jetpack.h	86
8.14 Platform.h	87
8.15 Player.h	87
8.16 Spring.h	88
8.17 StaticPlatform.h	89

8.18 TemporaryPlatform.h	89
8.19 VerticalPlatform.h	90
8.20 Observer.h	90
8.21 Score.h	90
8.22 Settings.h	91
8.23 Subject.h	92
8.24 Camera.h	93
8.25 Exception.h	93
8.26 Random.h	94
8.27 Stopwatch.h	94
8.28 Utilities.h	95
8.29 World.h	96
8.30 ConcreteFactory.h	96
8.31 Game.h	97
8.32 Resourcemanager.h	98
8.33 BackgroundView.h	99
8.34 BonusView.h	100
8.35 IView.h	100
8.36 PlatformView.h	101
8.37 PlayerView.h	102
8.38 ScoreView.h	102
Index	105

Chapter 1

Advanced-Programming-DoodleJump

Doodle Jump game implemented with SFML for the Advanced Programming course at the University of Antwerp

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Model	Namespace holds all Models	11
Observer	Namespace holds Observer pattern	12
Settings	Namespace for Settings used in World	12
Utils	Namespace holds all Utilities	13

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Model::AbstractFactory	15
View::ConcreteFactory	26
Model::CollisionBox	25
std::enable_shared_from_this	
View::IView	41
View::BackgroundView	16
View::BonusView	19
View::PlatformView	52
View::PlayerView	56
View::ScoreView	61
Event	35
CollisionEvent	25
DrawEvent	28
KeyPressedEvent	45
MoveEvent	46
NewDifficultyEvent	47
NewMaxHeightEvent	48
OutOfViewEvent	49
StopBonusEvent	64
std::exception	
Utils::Exception	35
Utils::FileException	36
Game	36
HighScore	37
HighScoreScore	37
IEventHandler	41
Controller::IController	39
Controller::BonusController	19
Controller::PlatformController	51
Controller::PlayerController	55
Model::Player	53
Model::Score	59
View::IView	41
Visitor::IVisitor	43

Model::Entity	29
Model::Background	15
Model::Bonus	17
Model::HorizontalPlatform	38
Model::Jetpack	44
Model::Platform	50
Model::Player	53
Model::Score	59
Model::Spring	62
Model::StaticPlatform	63
Model::TemporaryPlatform	69
Model::VerticalPlatform	71
Observer::Observer	48
Controller::IController	39
Model::Player	53
Model::Score	59
View::IView	41
Utils::Random	57
Utils::Resourceholder< Type >	58
Utils::Resourcemanager	59
Utils::Stopwatch	65
Observer::Subject	67
Model::Entity	29
Utils::Camera	20
Utils::Utilities	70
World	72

Chapter 4

Class Index

4.1 Class List

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

Model::AbstractFactory	15
Model::Background	15
View::BackgroundView	16
Model::Bonus	17
Controller::BonusController	19
View::BonusView	19
Utils::Camera	
Class for Camera	20
Model::CollisionBox	25
CollisionEvent	25
View::ConcreteFactory	26
DrawEvent	28
Model::Entity	
Class for Entity object	29
Event	35
Utils::Exception	35
Utils::FileException	36
Game	
Class for Game	36
HighScore	37
HighScoreScore	37
Model::HorizontalPlatform	38
Controller::IController	39
IEventHandler	41
View::IView	41
Visitor::IVisitor	43
Model::Jetpack	44
KeyPressedEvent	45
MoveEvent	46
NewDifficultyEvent	47
NewMaxHeightEvent	48
Observer::Observer	
Class for Observer of Observer pattern	48
OutOfViewEvent	49
Model::Platform	50

Controller::PlatformController	51
View::PlatformView	52
Model::Player	
Class for Player object, derives from Entity	53
Controller::PlayerController	55
View::PlayerView	56
Utils::Random	
Class for Random	57
Utils::Resourceholder< Type >	58
Utils::Resourcemanager	59
Model::Score	59
View::ScoreView	61
Model::Spring	62
Model::StaticPlatform	63
StopBonusEvent	64
Utils::Stopwatch	
Class for Stopwatch	65
Observer::Subject	
Class for Subject / Observable of observer pattern	67
Model::TemporaryPlatform	69
Utils::Utilities	
Class for Utilities	70
Model::VerticalPlatform	71
World	
Class for World , holds all the entities and is used for the overall game logic	72

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/AbstractFactory.h	77
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h	79
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/HighScore.h	82
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/IVisitor.h	83
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Observer.h	90
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Score.h	90
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Settings.h	91
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Subject.h	92
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/World.h	96
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/BonusController.h	77
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/IController.h	78
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/PlatformController.h	78
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/PlayerController.h	79
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Background.h	83
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Bonus.h	83
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Entity.h	84
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/HorizontalPlatform.h	85
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Jetpack.h	86
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Platform.h	87
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Player.h	87
/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Spring.h	88

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/[StaticPlatform.h](#)
89

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/[TemporaryPlatform.h](#)
89

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/[VerticalPlatform.h](#)
90

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/[Camera.h](#)
93

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/[Exception.h](#)
93

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/[Random.h](#)
94

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/[Stopwatch.h](#)
94

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/[Utilities.h](#)
95

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/[ConcreteFactory.h](#)
96

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/[Game.h](#) . 97

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/util/[ResourceManager.h](#)
98

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/[BackgroundView.h](#)
99

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/[BonusView.h](#)
100

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/[IView.h](#)
100

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/[PlatformView.h](#)
101

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/[PlayerView.h](#)
102

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/[ScoreView.h](#)
102

Chapter 6

Namespace Documentation

6.1 Model Namespace Reference

Namespace holds all Models.

Classes

- class [AbstractFactory](#)
- class [Background](#)
- class [Bonus](#)
- class [CollisionBox](#)
- class [Entity](#)

Class for [Entity](#) object.

- class [HorizontalPlatform](#)
- class [Jetpack](#)
- class [Platform](#)
- class [Player](#)

Class for [Player](#) object, derives from [Entity](#).

- class [Score](#)
- class [Spring](#)
- class [StaticPlatform](#)
- class [TemporaryPlatform](#)
- class [VerticalPlatform](#)

Enumerations

- enum [Type](#) {
 ePlayer = 0 , **eBonus** = 1 , **eStatic** = 2 , **eHorizontal** = 3 ,
 eVertical = 4 , **eTemporary** = 5 , **eBackground** = 6 , **eJetpack** = 7 ,
 eSpring = 8 , **eScore** = 9 }

6.1.1 Detailed Description

Namespace holds all Models.

Namespace holds all model.

6.1.2 Enumeration Type Documentation

6.1.2.1 Type

enum `Model::Type`

@Brief Enum containing enumerated Entities

6.2 Observer Namespace Reference

Namespace holds [Observer](#) pattern.

Classes

- class [Observer](#)
Class for [Observer](#) of [Observer](#) pattern.
- class [Subject](#)
Class for [Subject](#) / [Observable](#) of observer pattern.

6.2.1 Detailed Description

Namespace holds [Observer](#) pattern.

Namespace holds observer pattern.

6.3 Settings Namespace Reference

Namespace for [Settings](#) used in [World](#).

Enumerations

- enum [Difficulty](#) {
 eEasy = 0 , **eNormal** , **eDifficult** , **eHard** ,
 eExtreme }
Enum holding different difficulties.

6.3.1 Detailed Description

Namespace for [Settings](#) used in [World](#).

6.3.2 Enumeration Type Documentation

6.3.2.1 Difficulty

```
enum Settings::Difficulty
```

Enum holding different difficulties.

Difficulty

6.4 Utils Namespace Reference

Namespace holds all [Utilities](#).

Classes

- class [Camera](#)
Class for [Camera](#).
- class [Exception](#)
- class [FileException](#)
- class [Random](#)
Class for [Random](#).
- struct [Resourceholder](#)
- class [Resourcemanager](#)
- class [Stopwatch](#)
Class for [Stopwatch](#).
- class [Utilities](#)
Class for [Utilities](#).

6.4.1 Detailed Description

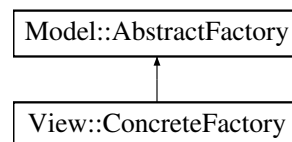
Namespace holds all [Utilities](#).

Chapter 7

Class Documentation

7.1 Model::AbstractFactory Class Reference

Inheritance diagram for Model::AbstractFactory:



Public Member Functions

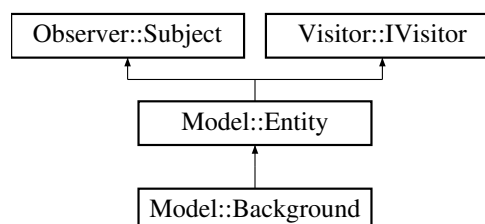
- virtual std::shared_ptr< [Model::Player](#) > **createPlayer** ()=0
- virtual std::shared_ptr< [Model::Entity](#) > **createStaticPlatform** ()=0
- virtual std::shared_ptr< [Model::Entity](#) > **createHorizontalPlatform** ()=0
- virtual std::shared_ptr< [Model::Entity](#) > **createVerticalPlatform** ()=0
- virtual std::shared_ptr< [Model::Entity](#) > **createTemporaryPlatform** ()=0
- virtual std::shared_ptr< [Model::Entity](#) > **createSpring** ()=0
- virtual std::shared_ptr< [Model::Entity](#) > **createJetpack** ()=0
- virtual std::shared_ptr< [Model::Entity](#) > **createBackground** ()=0
- virtual std::shared_ptr< [Model::Score](#) > **createScore** ()=0

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/AbstractFactory.h

7.2 Model::Background Class Reference

Inheritance diagram for Model::Background:



Public Member Functions

- `Model::Type getType ()` const override
Get type of *Entity* object.
- void `move` (bool collision) override
move *Entity* object

Additional Inherited Members

7.2.1 Member Function Documentation

7.2.1.1 getType()

`Model::Type` `Model::Background::getType ()` const [inline], [override], [virtual]

Get type of *Entity* object.

Returns

`Model::Type`

Implements `Model::Entity`.

7.2.1.2 move()

`void` `Model::Background::move (`
 `bool collision)` [inline], [override], [virtual]

move *Entity* object

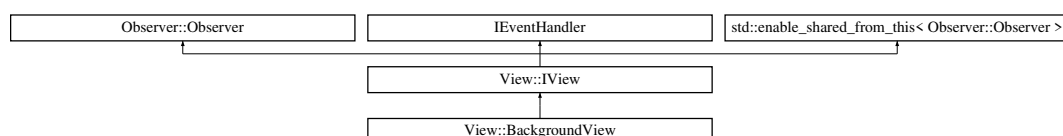
Implements `Model::Entity`.

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

- `/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Background.h`

7.3 View::BackgroundView Class Reference

Inheritance diagram for `View::BackgroundView`:



Public Member Functions

- **BackgroundView** (const std::shared_ptr< [Model::Entity](#) > &entity, const std::shared_ptr< sf::RenderWindow > &window)
- void [handleEvent](#) (const [DrawEvent](#) &event) override
- void [handleEvent](#) (const [OutOfViewEvent](#) &event) override

Additional Inherited Members

7.3.1 Member Function Documentation

7.3.1.1 [handleEvent\(\)](#) [1/2]

```
void BackgroundView::handleEvent (
    const DrawEvent & event ) [override], [virtual]
```

Reimplemented from [IEventHandler](#).

7.3.1.2 [handleEvent\(\)](#) [2/2]

```
void BackgroundView::handleEvent (
    const OutOfViewEvent & event ) [override], [virtual]
```

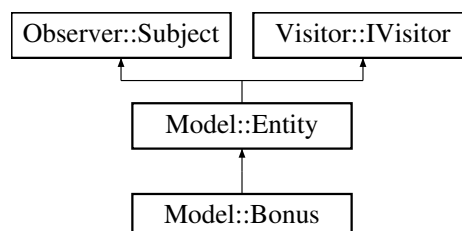
Reimplemented from [IEventHandler](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/BackgroundView.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/src/view/BackgroundView.cpp

7.4 Model::Bonus Class Reference

Inheritance diagram for Model::Bonus:



Public Types

- enum [Sort](#) { **eJetpack** = 0 , **eSpring** = 1 }
- Enum containing enumerated Bonuses.*

Public Member Functions

- [Model::Type](#) [getType](#) () const override
Get type of [Entity](#) object.
- [Sort](#) [getMSort](#) () const
Get sort of [Bonus](#) object.
- void [setMSort](#) ([Sort](#) sort)
Set sort of [Bonus](#) object.

Additional Inherited Members

7.4.1 Member Function Documentation

7.4.1.1 [getMSort\(\)](#)

[Bonus::Sort](#) [Bonus::getMSort](#) () const

Get sort of [Bonus](#) object.

Returns

7.4.1.2 [getType\(\)](#)

[Model::Type](#) [Model::Bonus::getType](#) () const [inline], [override], [virtual]

Get type of [Entity](#) object.

Returns

[Model::Type](#)

Implements [Model::Entity](#).

7.4.1.3 [setMSort\(\)](#)

void [Bonus::setMSort](#) (
 [Bonus::Sort](#) sort)

Set sort of [Bonus](#) object.

Parameters

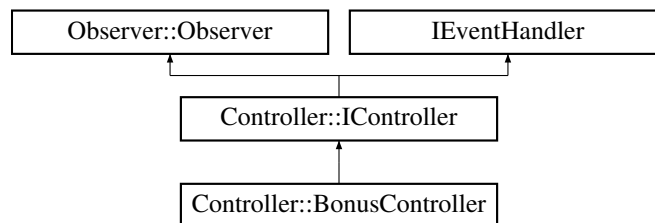
<i>sort</i>	Bonus::Sort
-------------	-----------------------------

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Bonus.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/model/Bonus.cpp

7.5 Controller::BonusController Class Reference

Inheritance diagram for Controller::BonusController:



Public Member Functions

- **BonusController** (std::shared_ptr< [Model::Entity](#) > &entity)

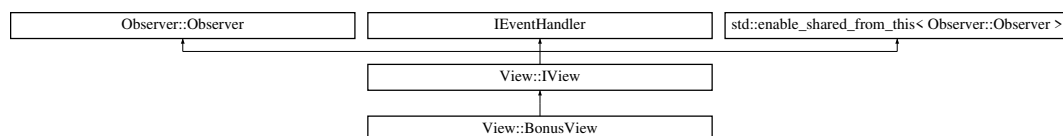
Additional Inherited Members

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/BonusController.h

7.6 View::BonusView Class Reference

Inheritance diagram for View::BonusView:



Public Member Functions

- **BonusView** (const std::shared_ptr< [Model::Entity](#) > &entity, const std::shared_ptr< sf::RenderWindow > &window)

Additional Inherited Members

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

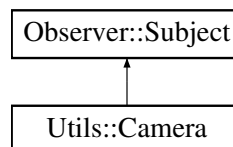
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/BonusView.h

7.7 Utils::Camera Class Reference

Class for [Camera](#).

```
#include <Camera.h>
```

Inheritance diagram for Utils::Camera:



Public Member Functions

- virtual **~Camera** ()=default
Default destructor.
- **Camera** (const [Camera](#) &)=delete
Deleted copy constructor.
- [Camera](#) & **operator=** (const [Camera](#) &)=delete
Deleted assignment operator.
- void **reset** ()
- std::pair< float, float > [getWorldDimensions](#) () const
Get world dimensions.
- void [setWorldDimensions](#) (float right, float top, float left=0.f, float bottom=0.f)
Set world dimensions.
- std::pair< float, float > [getWindowDimensions](#) () const
Get window dimensions.
- void [setWindowDimensions](#) (float right, float bottom, float left=0.f, float top=0.f)
Set viewport / window dimensions.
- std::pair< float, float > [transform](#) (float x, float y, float left=0.f, float top=0.f) const
Transforms world coordinates to viewport / window coordinates.
- std::pair< float, float > [inverseTransform](#) (float x, float y) const
Transforms viewport / window coordinates to world coordinates.
- void [move](#) (float x, float y)

- Move [Camera](#).
- float [getX](#) () const
Get x coordinate of [Camera](#).
- float [getY](#) () const
Get y coordinate of [Camera](#).
- float [getMaxHeight](#) () const
Get maximum height of [Camera](#).
- float [getLastMaxHeight](#) () const
Get last maximum height of [Camera](#).
- bool [isMaxHeight](#) (float height)
Check if given height is greater or equal to current maximum height.

Static Public Member Functions

- static [Camera](#) & [getInstance](#) ()
Get instance of [Camera](#) object.

7.7.1 Detailed Description

Class for [Camera](#).

7.7.2 Member Function Documentation

7.7.2.1 [getInstance\(\)](#)

```
Camera & Camera::getInstance ( ) [static]
```

Get instance of [Camera](#) object.

Returns

[Camera](#)

7.7.2.2 [getMaxHeight\(\)](#)

```
float Utils::Camera::getMaxHeight ( ) const [inline]
```

Get maximum height of [Camera](#).

Returns

float

7.7.2.3 getWindowDimensions()

```
std::pair< float, float > Camera::getWindowDimensions ( ) const
```

Get window dimensions.

Returns

7.7.2.4 getWorldDimensions()

```
std::pair< float, float > Camera::getWorldDimensions ( ) const
```

Get world dimensions.

Returns

`std::pair<float, float>`

7.7.2.5 getX()

```
float Utils::Camera::getX ( ) const [inline]
```

Get x coordinate of [Camera](#).

Returns

`float`

7.7.2.6 getY()

```
float Utils::Camera::getY ( ) const [inline]
```

Get y coordinate of [Camera](#).

Returns

`float`

7.7.2.7 inverseTransform()

```
std::pair< float, float > Camera::inverseTransform (
    float x,
    float y ) const
```

Transforms viewport / window coordinates to world coordinates.

Parameters

<i>x</i>	float
<i>y</i>	float

Returns

std::pair<float, float>

7.7.2.8 isMaxHeight()

```
bool Camera::isMaxHeight (
    float height )
```

Check if given height is greater or equal to current maximum height.

Parameters

<i>height</i>	float
---------------	-------

Returns

bool

7.7.2.9 move()

```
void Camera::move (
    float x,
    float y )
```

Move [Camera](#).

Parameters

<i>x</i>	float
<i>y</i>	float

7.7.2.10 operator=()

```
Camera & Utils::Camera::operator= (
    const Camera & ) [delete]
```

Deleted assignment operator.

Returns

[Camera](#)**7.7.2.11 setWindowDimensions()**

```
void Camera::setWindowDimensions (
    float right,
    float bottom,
    float left = 0.f,
    float top = 0.f )
```

Set viewport / window dimensions.

Parameters

<i>right</i>	float
<i>bottom</i>	float
<i>left</i>	float
<i>top</i>	float

7.7.2.12 setWorldDimensions()

```
void Camera::setWorldDimensions (
    float right,
    float top,
    float left = 0.f,
    float bottom = 0.f )
```

Set world dimensions.

Parameters

<i>right</i>	float
<i>top</i>	float
<i>left</i>	float
<i>bottom</i>	float

7.7.2.13 transform()

```
std::pair< float, float > Camera::transform (
    float x,
    float y,
```

```
float left = 0.f,
float top = 0.f ) const
```

Transforms world coordinates to viewport / window coordinates.

Parameters

<i>x</i>	float
<i>y</i>	float

Returns

std::pair<float, float>

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/Camera.↔
h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/util/Camera.cpp

7.8 Model::CollisionBox Class Reference

Public Member Functions

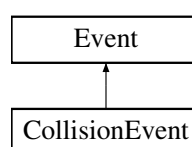
- **CollisionBox** (float left, float width, float bottom, float height)
- float **getLeft** () const
- float **getWidth** () const
- float **getBottom** () const
- float **getHeight** () const

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Entity.↔
h

7.9 CollisionEvent Class Reference

Inheritance diagram for CollisionEvent:



Public Member Functions

- **CollisionEvent** (std::shared_ptr< [Model::Entity](#) > entity, std::shared_ptr< [Model::Player](#) > player)
- void [send](#) ([IEventHandler](#) &handler) const override
- const std::shared_ptr< [Model::Entity](#) > & **getEntity** () const
- const std::shared_ptr< [Model::Player](#) > & **getPlayer** () const

Additional Inherited Members

7.9.1 Member Function Documentation

7.9.1.1 send()

```
void CollisionEvent::send (
    IEventHandler & handler ) const [inline], [override], [virtual]
```

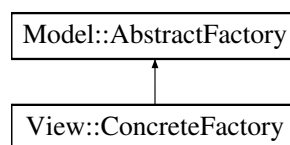
Reimplemented from [Event](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h

7.10 View::ConcreteFactory Class Reference

Inheritance diagram for View::ConcreteFactory:



Public Member Functions

- **ConcreteFactory** (const std::shared_ptr< sf::RenderWindow > &window)
- std::shared_ptr< [Model::Player](#) > [createPlayer](#) () override
- std::shared_ptr< [Model::Entity](#) > [createStaticPlatform](#) () override
- std::shared_ptr< [Model::Entity](#) > [createHorizontalPlatform](#) () override
- std::shared_ptr< [Model::Entity](#) > [createVerticalPlatform](#) () override
- std::shared_ptr< [Model::Entity](#) > [createTemporaryPlatform](#) () override
- std::shared_ptr< [Model::Entity](#) > [createSpring](#) () override
- std::shared_ptr< [Model::Entity](#) > [createJetpack](#) () override
- std::shared_ptr< [Model::Entity](#) > [createBackground](#) () override
- std::shared_ptr< [Model::Score](#) > [createScore](#) () override

7.10.1 Member Function Documentation

7.10.1.1 createBackground()

```
std::shared_ptr< Model::Entity > ConcreteFactory::createBackground ( ) [override], [virtual]
```

Implements [Model::AbstractFactory](#).

7.10.1.2 createHorizontalPlatform()

```
std::shared_ptr< Model::Entity > ConcreteFactory::createHorizontalPlatform ( ) [override],  
[virtual]
```

Implements [Model::AbstractFactory](#).

7.10.1.3 createJetpack()

```
std::shared_ptr< Model::Entity > ConcreteFactory::createJetpack ( ) [override], [virtual]
```

Implements [Model::AbstractFactory](#).

7.10.1.4 createPlayer()

```
std::shared_ptr< Model::Player > ConcreteFactory::createPlayer ( ) [override], [virtual]
```

Implements [Model::AbstractFactory](#).

7.10.1.5 createScore()

```
std::shared_ptr< Model::Score > ConcreteFactory::createScore ( ) [override], [virtual]
```

Implements [Model::AbstractFactory](#).

7.10.1.6 createSpring()

```
std::shared_ptr< Model::Entity > ConcreteFactory::createSpring ( ) [override], [virtual]
```

Implements [Model::AbstractFactory](#).

7.10.1.7 createStaticPlatform()

```
std::shared_ptr< Model::Entity > ConcreteFactory::createStaticPlatform ( ) [override], [virtual]
```

Implements [Model::AbstractFactory](#).

7.10.1.8 createTemporaryPlatform()

```
std::shared_ptr< Model::Entity > ConcreteFactory::createTemporaryPlatform ( ) [override],  
[virtual]
```

Implements [Model::AbstractFactory](#).

7.10.1.9 createVerticalPlatform()

```
std::shared_ptr< Model::Entity > ConcreteFactory::createVerticalPlatform ( ) [override],  
[virtual]
```

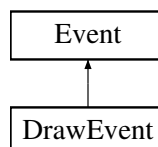
Implements [Model::AbstractFactory](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/Concrete↵
Factory.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/src/Concrete↵
Factory.cpp

7.11 DrawEvent Class Reference

Inheritance diagram for DrawEvent:



Public Member Functions

- void [send](#) (IEventHandler &handler) const override

Additional Inherited Members

7.11.1 Member Function Documentation

7.11.1.1 send()

```
void DrawEvent::send (
    IEventHandler & handler ) const [inline], [override], [virtual]
```

Reimplemented from [Event](#).

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

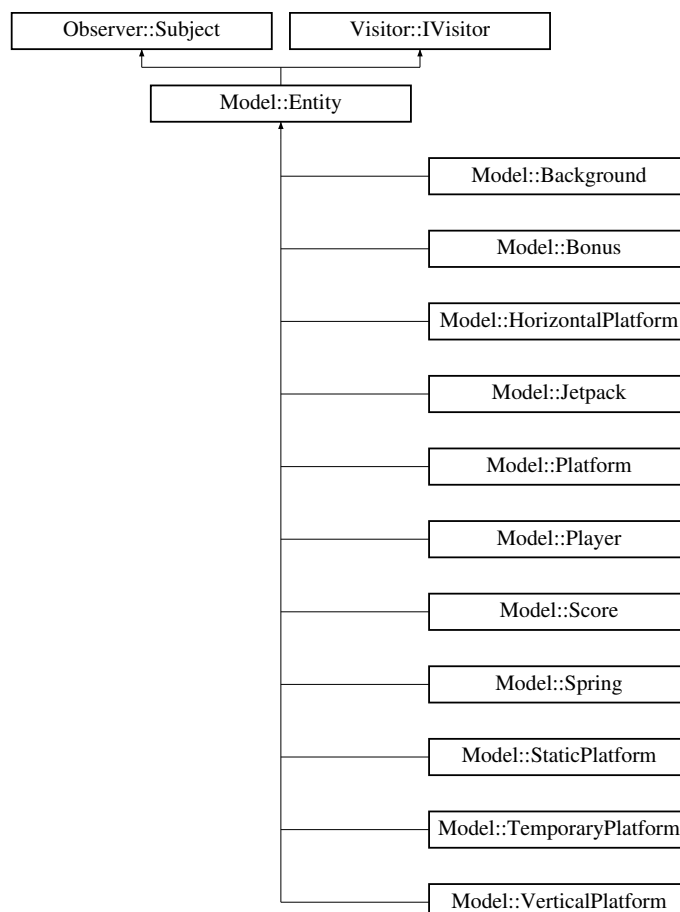
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h

7.12 Model::Entity Class Reference

Class for [Entity](#) object.

```
#include <Entity.h>
```

Inheritance diagram for Model::Entity:



Public Member Functions

- **Entity** ()
Default constructor.
- **Entity** (unsigned int score, float spawnRate)
- virtual **~Entity** ()=default
Default constructor.
- float **getX** () const
Get the x value of [Entity](#) object.
- float **getY** () const
Get the y value of [Entity](#) object.
- void **setX** (float x)
Set the x value of [Entity](#) object.
- void **setY** (float y)
Set the y value of [Entity](#) object.
- float **getWidth** () const
- float **getHeight** () const
Get height of [Entity](#) object.
- void **setWidth** (float width)
Set width of [Entity](#) object.
- void **setHeight** (float height)
Set height of [Entity](#) object.
- virtual void **move** (bool collision)=0
move [Entity](#) object
- void **move** (float x, float y)
Move [Entity](#) object in x and y direction.
- virtual **Model::Type** **getType** () const =0
Get type of [Entity](#) object.
- virtual void **onDestroy** ()
On destroy event of [Entity](#) function will be executed.
- void **visit** (**Model::Player** &player) override
- virtual void **accept** (const std::shared_ptr< **Visitor::IVisitor** > &visitor)
- void **setRemoveFlag** (bool flag)
- virtual bool **getRemovable** () const
- virtual bool **isRemovable** () const
- virtual bool **isBonus** () const
- void **setScore** (unsigned int score)
- virtual unsigned int **getScore** () const
- virtual float **getSpawnRate** () const
- void **setSpawnRate** (float spawnRate)

Protected Attributes

- float **mX**
- float **mY**
- float **mWidth** {}
- float **mHeight** {}
- bool **mRemoveFlag**
- unsigned int **mScore**
- float **mSpawnRate**

7.12.1 Detailed Description

Class for [Entity](#) object.

7.12.2 Member Function Documentation

7.12.2.1 getHeight()

```
float Entity::getHeight ( ) const
```

Get height of [Entity](#) object.

Returns

float

7.12.2.2 getType()

```
virtual Model::Type Model::Entity::getType ( ) const [pure virtual]
```

Get type of [Entity](#) object.

Returns

[Model::Type](#)

Implemented in [Model::Background](#), [Model::Bonus](#), [Model::HorizontalPlatform](#), [Model::Jetpack](#), [Model::Platform](#), [Model::Player](#), [Model::Spring](#), [Model::StaticPlatform](#), [Model::TemporaryPlatform](#), [Model::VerticalPlatform](#), and [Model::Score](#).

7.12.2.3 getX()

```
float Entity::getX ( ) const
```

Get the x value of [Entity](#) object.

Returns

float

7.12.2.4 `getY()`

```
float Entity::getY ( ) const
```

Get the y value of [Entity](#) object.

Returns

float

7.12.2.5 `move()` [1/2]

```
virtual void Model::Entity::move (
    bool collision ) [pure virtual]
```

move [Entity](#) object

Implemented in [Model::Background](#), [Model::HorizontalPlatform](#), [Model::Jetpack](#), [Model::Platform](#), [Model::Player](#), [Model::Spring](#), [Model::StaticPlatform](#), [Model::TemporaryPlatform](#), [Model::VerticalPlatform](#), and [Model::Score](#).

7.12.2.6 `move()` [2/2]

```
void Entity::move (
    float x,
    float y )
```

Move [Entity](#) object in x and y direction.

Parameters

<i>x</i>	float, added to mX
<i>y</i>	float, added to mY

7.12.2.7 `setHeight()`

```
void Entity::setHeight (
    float height )
```

Set height of [Entity](#) object.

Parameters

<i>height</i>	float
---------------	-------

7.12.2.8 setWidth()

```
void Entity::setWidth (
    float width )
```

Set width of [Entity](#) object.

Parameters

<i>width</i>	float
--------------	-------

7.12.2.9 setX()

```
void Entity::setX (
    float x )
```

Set the x value of [Entity](#) object.

Parameters

<i>x</i>	float
----------	-------

7.12.2.10 setY()

```
void Entity::setY (
    float y )
```

Set the y value of [Entity](#) object.

Parameters

<i>y</i>	float
----------	-------

7.12.2.11 visit()

```
void Model::Entity::visit (
    Model::Player & player ) [inline], [override], [virtual]
```

Implements [Visitor::IVisitor](#).

7.12.3 Member Data Documentation

7.12.3.1 mHeight

```
float Model::Entity::mHeight {} [protected]
```

Height of [Entity](#) collision box

7.12.3.2 mWidth

```
float Model::Entity::mWidth {} [protected]
```

Width of [Entity](#) collision box

7.12.3.3 mX

```
float Model::Entity::mX [protected]
```

X-coordinate of [Entity](#)

7.12.3.4 mY

```
float Model::Entity::mY [protected]
```

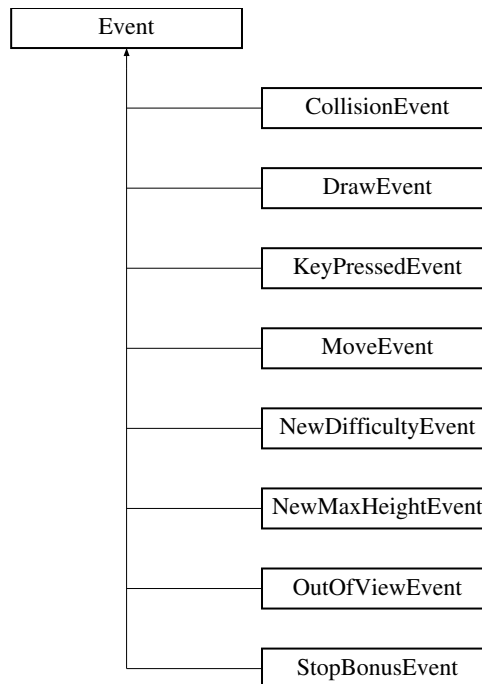
Y-coordinate of [Entity](#)

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

- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Entity.h](#)↔
- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/model/Entity.cpp](#)

7.13 Event Class Reference

Inheritance diagram for Event:



Public Member Functions

- **Event** (EventType mEvent)
- virtual void **send** ([IEventHandler](#) &handler) const
- EventType **getEvent** () const

Protected Attributes

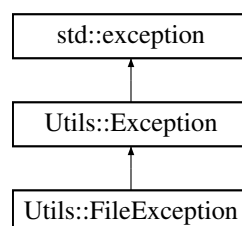
- EventType **mEvent**

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h

7.14 Utils::Exception Class Reference

Inheritance diagram for Utils::Exception:



Public Member Functions

- **Exception** (std::string value)
- const char * **what** () const noexcept override

Protected Attributes

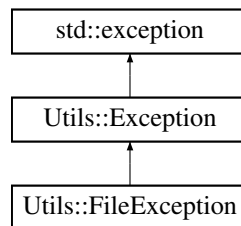
- std::string **mValue**

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/Exception.h↔

7.15 Utils::FileException Class Reference

Inheritance diagram for Utils::FileException:



Public Member Functions

- **FileException** (std::string file, std::string sort)

Additional Inherited Members

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/Exception.h↔

7.16 Game Class Reference

Class for [Game](#).

```
#include <Game.h>
```

Public Member Functions

- **Game** (unsigned int width=800, unsigned int height=1440)
- void **initializeResources** ()
- void **processEvents** ()
- void **handlePlayerInput** (sf::Keyboard::Key key, bool isPressed)
- void **render** ()
- void **run** ()
- void **drawHighScoreTable** ()

7.16.1 Detailed Description

Class for [Game](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/Game.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/src/Game.cpp

7.17 HighScore Class Reference

Public Member Functions

- **HighScore** (const [HighScore](#) &)=delete
- **HighScore** & **operator=** (const [HighScore](#) &)=delete
- void **load** ()
- void **save** ()
- void **add** (const std::shared_ptr< [HighScoreScore](#) > &score)
- const std::vector< std::shared_ptr< [HighScoreScore](#) > > & **getScores** () const
- unsigned int **getHighScore** () const

Static Public Member Functions

- static [HighScore](#) & **getInstance** ()

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/HighScore.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/HighScore.cpp

7.18 HighScoreScore Struct Reference

Public Member Functions

- **HighScoreScore** (unsigned int score, std::string name)
- std::string **toString** () const

Public Attributes

- unsigned int **mScore**
- std::string **mName**

Friends

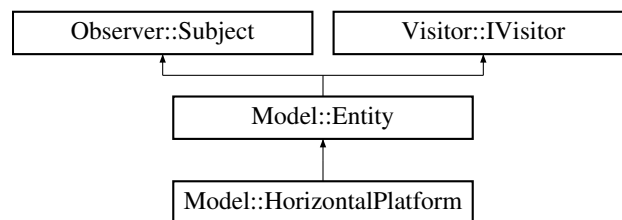
- std::ostream & **operator**<< (std::ostream &os, const [HighScoreScore](#) &score)

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/HighScore.h

7.19 Model::HorizontalPlatform Class Reference

Inheritance diagram for Model::HorizontalPlatform:



Public Member Functions

- [Model::Type](#) **getType** () const override
Get type of [Entity](#) object.
- void [move](#) (bool collision) override
move [Entity](#) object
- void **initBounds** ()

Additional Inherited Members

7.19.1 Member Function Documentation

7.19.1.1 `getType()`

```
Model::Type HorizontalPlatform::getType ( ) const [override], [virtual]
```

Get type of [Entity](#) object.

Returns

[Model::Type](#)

Implements [Model::Entity](#).

7.19.1.2 `move()`

```
void HorizontalPlatform::move (
    bool collision ) [override], [virtual]
```

move [Entity](#) object

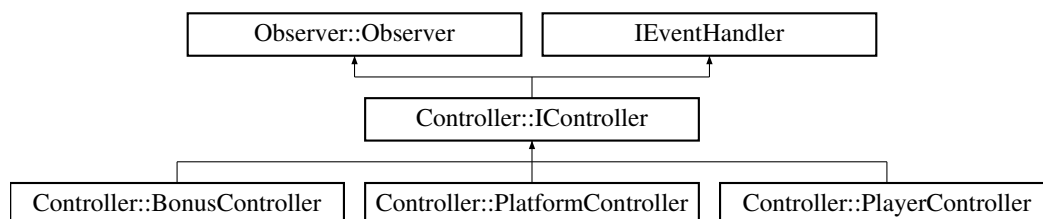
Implements [Model::Entity](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/HorizontalPlatform.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/model/HorizontalPlatform.cpp

7.20 Controller::IController Class Reference

Inheritance diagram for Controller::IController:



Public Member Functions

- **IController** (std::shared_ptr< [Model::Entity](#) > &entity)
- const std::shared_ptr< [Model::Entity](#) > & **getEntity** () const
- void **onTrigger** (EventType type, const std::shared_ptr< [Event](#) > &event) override
Perform operation on trigger from Subject.
- void **handleEvent** (const [KeyPressedEvent](#) &event) override
- void **handleEvent** (const [MoveEvent](#) &event) override
- void **handleEvent** (const [CollisionEvent](#) &event) override

Protected Attributes

- `std::shared_ptr< Model::Entity > mEntity`

7.20.1 Member Function Documentation

7.20.1.1 `handleEvent()` [1/3]

```
void Controller::IController::handleEvent (
    const CollisionEvent & event ) [inline], [override], [virtual]
```

Reimplemented from [IEventHandler](#).

7.20.1.2 `handleEvent()` [2/3]

```
void Controller::IController::handleEvent (
    const KeyPressedEvent & event ) [inline], [override], [virtual]
```

Reimplemented from [IEventHandler](#).

7.20.1.3 `handleEvent()` [3/3]

```
void IController::handleEvent (
    const MoveEvent & event ) [override], [virtual]
```

Reimplemented from [IEventHandler](#).

7.20.1.4 `onTrigger()`

```
void Controller::IController::onTrigger (
    EventType type,
    const std::shared_ptr< Event > & event ) [inline], [override], [virtual]
```

Perform operation on trigger from Subject.

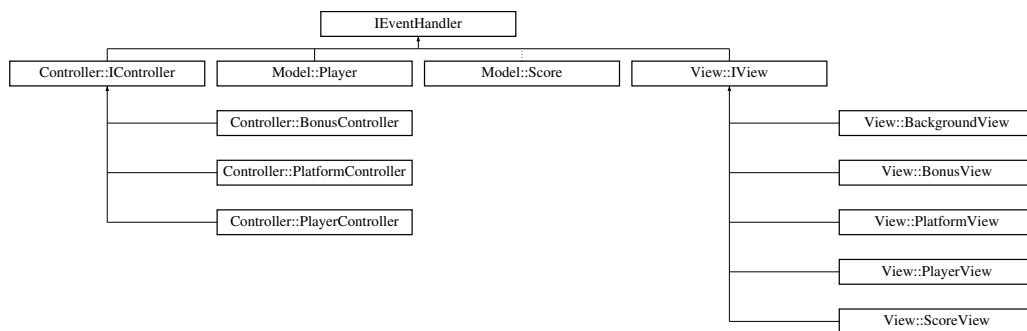
Implements [Observer::Observer](#).

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

- `/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/IController.h`↔
- `/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/controller/IController.cpp`↔

7.21 IEventHandler Class Reference

Inheritance diagram for IEventHandler:



Public Member Functions

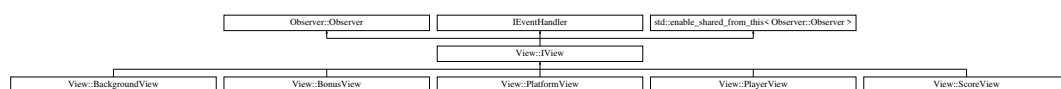
- virtual void **handleEvent** (const [Event](#) &event)
- virtual void **handleEvent** (const [DrawEvent](#) &event)
- virtual void **handleEvent** (const [OutOfViewEvent](#) &event)
- virtual void **handleEvent** (const [NewMaxHeightEvent](#) &event)
- virtual void **handleEvent** (const [CollisionEvent](#) &event)
- virtual void **handleEvent** (const [KeyPressedEvent](#) &event)
- virtual void **handleEvent** (const [MoveEvent](#) &event)
- virtual void **handleEvent** (const [StopBonusEvent](#) &event)
- virtual void **handleEvent** (const [NewDifficultyEvent](#) &event)

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

- `/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h`

7.22 View::IView Class Reference

Inheritance diagram for View::IView:



Public Member Functions

- **IView** (const std::shared_ptr< [Model::Entity](#) > &entity, const std::shared_ptr< sf::RenderWindow > &window)
- void **drawCollisionBox** ()
- void **onTrigger** (EventType type, const std::shared_ptr< [Event](#) > &event) override

Perform operation on trigger from Subject.
- void **handleEvent** (const [DrawEvent](#) &event) override
- void **handleEvent** (const [OutOfViewEvent](#) &event) override
- virtual void **handleEvent** (const [CollisionEvent](#) &event) override
- void **handleEvent** (const [NewDifficultyEvent](#) &event) override

Static Public Member Functions

- `template<class Type >`
static void **setRainbowColor** (const std::unique_ptr< sf::Text > &object)

Protected Attributes

- std::shared_ptr< [Model::Entity](#) > **mEntity**
- std::unique_ptr< sf::Sprite > **mSprite**
- std::shared_ptr< sf::RenderWindow > **mWindow**
- std::unique_ptr< sf::Sound > **mSound**

7.22.1 Member Function Documentation

7.22.1.1 `handleEvent()` [1/4]

```
virtual void View::IView::handleEvent (
    const CollisionEvent & event ) [inline], [override], [virtual]
```

Reimplemented from [IEventHandler](#).

7.22.1.2 `handleEvent()` [2/4]

```
void IView::handleEvent (
    const DrawEvent & event ) [override], [virtual]
```

Reimplemented from [IEventHandler](#).

7.22.1.3 `handleEvent()` [3/4]

```
void View::IView::handleEvent (
    const NewDifficultyEvent & event ) [inline], [override], [virtual]
```

Reimplemented from [IEventHandler](#).

7.22.1.4 `handleEvent()` [4/4]

```
void IView::handleEvent (
    const OutOfViewEvent & event ) [override], [virtual]
```

Reimplemented from [IEventHandler](#).

7.22.1.5 onTrigger()

```
void IView::onTrigger (
    EventType type,
    const std::shared_ptr< Event > & event ) [override], [virtual]
```

Perform operation on trigger from Subject.

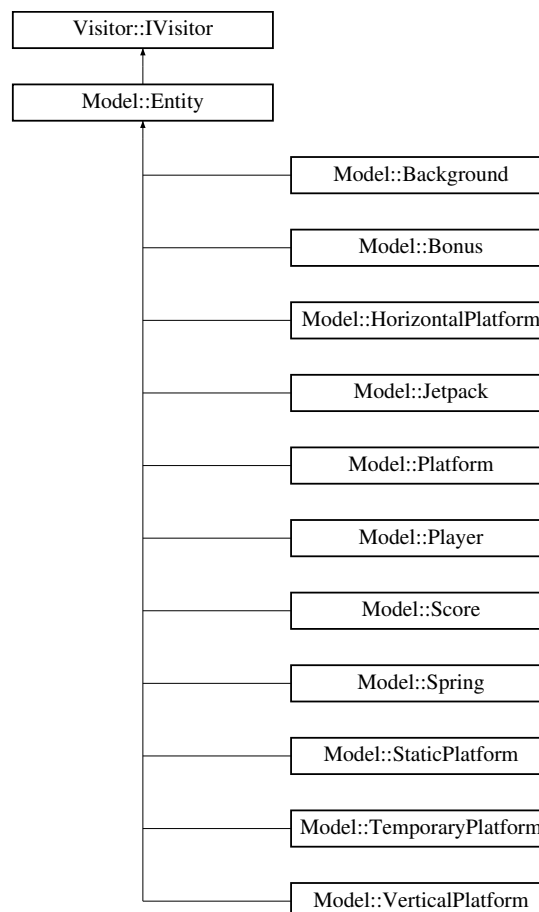
Implements [Observer::Observer](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/IView.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/src/view/IView.cpp

7.23 Visitor::IVisitor Class Reference

Inheritance diagram for Visitor::IVisitor:



Public Member Functions

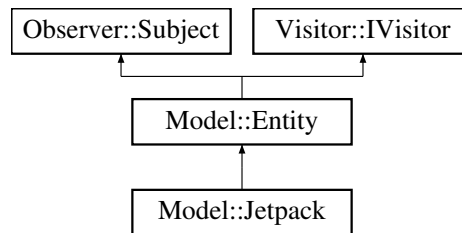
- virtual void **visit** ([Model::Player](#) &player)=0

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/IVisitor.h

7.24 Model::Jetpack Class Reference

Inheritance diagram for Model::Jetpack:



Public Member Functions

- **Jetpack** (bool started)
- `Model::Type` `getType` () const override
Get type of `Entity` object.
- void `move` (bool collision) override
move `Entity` object
- void `initBounds` ()
- void `visit` (`Model::Player` &player) override
- bool `isRemovable` () const override
- bool `isBonus` () const override

Public Attributes

- bool `mStarted`

Additional Inherited Members

7.24.1 Member Function Documentation

7.24.1.1 `getType()`

```
Model::Type Model::Jetpack::getType ( ) const [override], [virtual]
```

Get type of `Entity` object.

Returns

`Model::Type`

Implements `Model::Entity`.

7.24.1.2 isBonus()

```
bool Model::Jetpack::isBonus ( ) const [inline], [override], [virtual]
```

Reimplemented from [Model::Entity](#).

7.24.1.3 isRemovable()

```
bool Model::Jetpack::isRemovable ( ) const [inline], [override], [virtual]
```

Reimplemented from [Model::Entity](#).

7.24.1.4 move()

```
void Jetpack::move (
    bool collision ) [override], [virtual]
```

move [Entity](#) object

Implements [Model::Entity](#).

7.24.1.5 visit()

```
void Model::Jetpack::visit (
    Model::Player & player ) [inline], [override], [virtual]
```

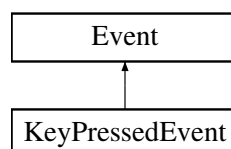
Reimplemented from [Model::Entity](#).

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

- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Jetpack.h](#)
- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/model/Jetpack.cpp](#)

7.25 KeyPressedEvent Class Reference

Inheritance diagram for KeyPressedEvent:



Public Member Functions

- **KeyPressedEvent** (std::string key, bool isPressed)
- void [send](#) ([IEventHandler](#) &handler) const override
- const std::string & **getKey** () const
- bool **isPressed** () const

Additional Inherited Members

7.25.1 Member Function Documentation

7.25.1.1 send()

```
void KeyPressedEvent::send (  
    IEventHandler & handler ) const [inline], [override], [virtual]
```

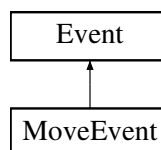
Reimplemented from [Event](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h

7.26 MoveEvent Class Reference

Inheritance diagram for MoveEvent:



Public Member Functions

- **MoveEvent** (bool collided)
- void [send](#) ([IEventHandler](#) &handler) const override
- bool **isCollided** () const

Additional Inherited Members

7.26.1 Member Function Documentation

7.26.1.1 send()

```
void MoveEvent::send (
    IEventHandler & handler ) const [inline], [override], [virtual]
```

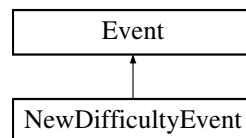
Reimplemented from [Event](#).

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

- `/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h`

7.27 NewDifficultyEvent Class Reference

Inheritance diagram for NewDifficultyEvent:



Public Member Functions

- **NewDifficultyEvent** ([Settings::Difficulty](#) difficulty)
- void [send](#) ([IEventHandler](#) &handler) const override
- [Settings::Difficulty](#) **getDifficulty** () const

Additional Inherited Members

7.27.1 Member Function Documentation

7.27.1.1 send()

```
void NewDifficultyEvent::send (
    IEventHandler & handler ) const [inline], [override], [virtual]
```

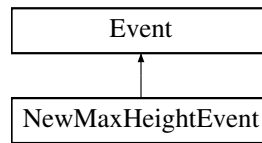
Reimplemented from [Event](#).

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

- `/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h`

7.28 NewMaxHeightEvent Class Reference

Inheritance diagram for NewMaxHeightEvent:



Public Member Functions

- **NewMaxHeightEvent** (float lastHeight, float newHeight)
- void **send** (IEventHandler &handler) const override
- float **getLastHeight** () const
- float **getNewHeight** () const

Additional Inherited Members

7.28.1 Member Function Documentation

7.28.1.1 send()

```
void NewMaxHeightEvent::send (
    IEventHandler & handler ) const [inline], [override], [virtual]
```

Reimplemented from [Event](#).

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

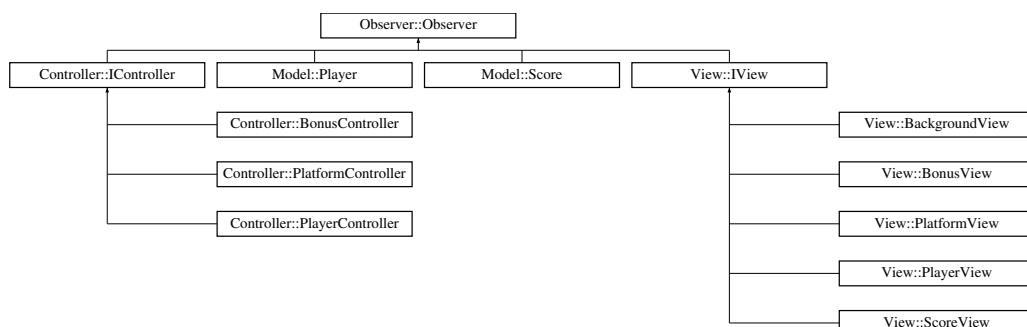
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h

7.29 Observer::Observer Class Reference

Class for [Observer](#) of [Observer](#) pattern.

```
#include <Observer.h>
```

Inheritance diagram for Observer::Observer:



Public Member Functions

- **Observer** ()=default
Default constructor.
- virtual **~Observer** ()=default
Default destructor.
- virtual void **onTrigger** (EventType type, const std::shared_ptr< **Event** > &event)=0
*Perform operation on trigger from **Subject**.*

7.29.1 Detailed Description

Class for **Observer** of **Observer** pattern.

7.29.2 Member Function Documentation

7.29.2.1 onTrigger()

```
virtual void Observer::Observer::onTrigger (
    EventType type,
    const std::shared_ptr< Event > & event ) [pure virtual]
```

Perform operation on trigger from **Subject**.

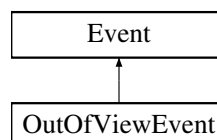
Implemented in **Controller::IController**, **Model::Player**, **Model::Score**, and **View::IView**.

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Observer.h

7.30 OutOfViewEvent Class Reference

Inheritance diagram for OutOfViewEvent:



Public Member Functions

- void **send** (**IEventHandler** &handler) const override

Additional Inherited Members

7.30.1 Member Function Documentation

7.30.1.1 send()

```
void OutOfViewEvent::send (
    IEventHandler & handler ) const [inline], [override], [virtual]
```

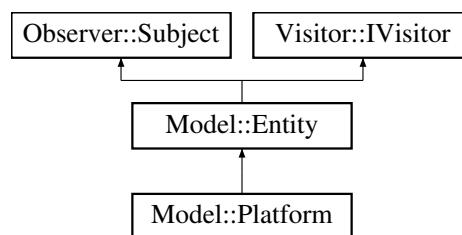
Reimplemented from [Event](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h

7.31 Model::Platform Class Reference

Inheritance diagram for Model::Platform:



Public Member Functions

- **Platform** ([Model::Type](#) sort)
- [Model::Type](#) **getType** () const override
Get type of [Entity](#) object.
- void [move](#) (bool collision) override
move [Entity](#) object
- void **initBounds** ()

Additional Inherited Members

7.31.1 Member Function Documentation

7.31.1.1 `getType()`

```
Model::Type Model::Platform::getType ( ) const [inline], [override], [virtual]
```

Get type of [Entity](#) object.

Returns

[Model::Type](#)

Implements [Model::Entity](#).

7.31.1.2 `move()`

```
void Platform::move (
    bool collision ) [override], [virtual]
```

move [Entity](#) object

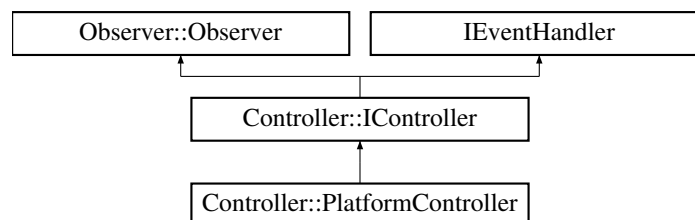
Implements [Model::Entity](#).

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

- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Platform.↵h](#)
- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/model/Platform.↵cpp](#)

7.32 **Controller::PlatformController Class Reference**

Inheritance diagram for Controller::PlatformController:

**Public Member Functions**

- **PlatformController** (std::shared_ptr< [Model::Entity](#) > &entity)

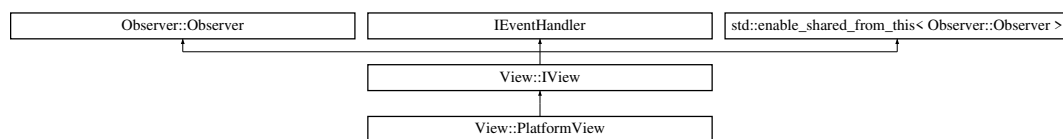
Additional Inherited Members

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

- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/PlatformController.h](#)

7.33 View::PlatformView Class Reference

Inheritance diagram for View::PlatformView:



Public Member Functions

- **PlatformView** (const std::shared_ptr< [Model::Entity](#) > &entity, const std::shared_ptr< sf::RenderWindow > &window)
- void [handleEvent](#) (const [CollisionEvent](#) &event) override

Additional Inherited Members

7.33.1 Member Function Documentation

7.33.1.1 handleEvent()

```
void View::PlatformView::handleEvent (
    const CollisionEvent & event ) [inline], [override], [virtual]
```

Reimplemented from [View::IView](#).

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

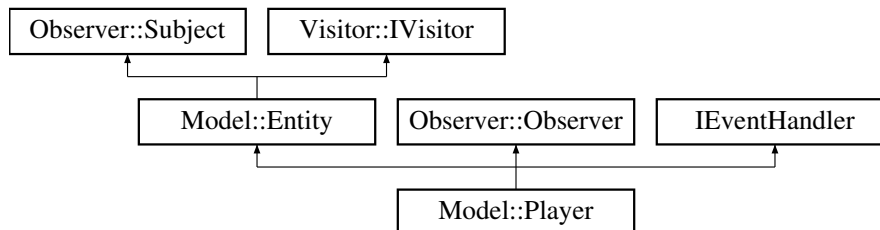
- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/PlatformView.h](#)

7.34 Model::Player Class Reference

Class for [Player](#) object, derives from [Entity](#).

```
#include <Player.h>
```

Inheritance diagram for Model::Player:



Public Member Functions

- [Model::Type](#) **getType** () const override
Get type of [Entity](#) object.
- void [move](#) (bool collision) override
move [Entity](#) object
- const std::pair< float, float > & **getVelocity** () const
- void **setVelocity** (const std::pair< float, float > &velocity)
- const std::pair< float, float > & **getDirection** () const
- void **setDirection** (const std::pair< float, float > &direction)
- const float **getMaxVelocity** () const
- const float **getMaxAcceleration** () const
- const float **getDrag** () const
- void **setDrag** (float drag)
- bool **isMovingUp** () const
- void **setIsMovingUp** (bool isMovingUp)
- bool **isMovingDown** () const
- void **setIsMovingDown** (bool isMovingDown)
- bool **isMovingLeft** () const
- void **setIsMovingLeft** (bool isMovingLeft)
- bool **isMovingRight** () const
- void **setIsMovingRight** (bool isMovingRight)
- void [accept](#) (const std::shared_ptr< [Visitor::IVisitor](#) > &visitor) override
- void [onTrigger](#) (EventType type, const std::shared_ptr< [Event](#) > &event) override
Perform operation on trigger from Subject.
- void [handleEvent](#) (const [StopBonusEvent](#) &event) override

Additional Inherited Members

7.34.1 Detailed Description

Class for [Player](#) object, derives from [Entity](#).

7.34.2 Member Function Documentation

7.34.2.1 accept()

```
void Model::Player::accept (
    const std::shared_ptr< Visitor::IVisitor > & visitor ) [inline], [override],
[virtual]
```

Reimplemented from [Model::Entity](#).

7.34.2.2 getType()

```
Model::Type Model::Player::getType ( ) const [inline], [override], [virtual]
```

Get type of [Entity](#) object.

Returns

[Model::Type](#)

Implements [Model::Entity](#).

7.34.2.3 handleEvent()

```
void Model::Player::handleEvent (
    const StopBonusEvent & event ) [inline], [override], [virtual]
```

Reimplemented from [IEventHandler](#).

7.34.2.4 move()

```
void Player::move (
    bool collision ) [override], [virtual]
```

move [Entity](#) object

Implements [Model::Entity](#).

7.34.2.5 onTrigger()

```
void Model::Player::onTrigger (
    EventType type,
    const std::shared_ptr< Event > & event ) [inline], [override], [virtual]
```

Perform operation on trigger from Subject.

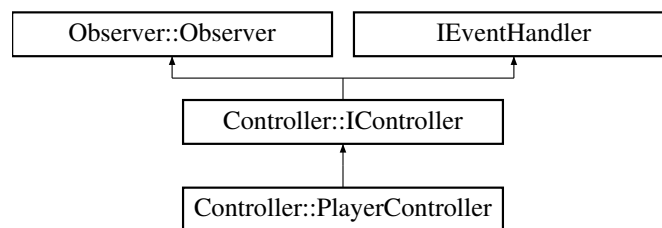
Implements [Observer::Observer](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Player.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/model/Player.cpp

7.35 Controller::PlayerController Class Reference

Inheritance diagram for Controller::PlayerController:



Public Member Functions

- **PlayerController** (std::shared_ptr< [Model::Entity](#) > &entity)
- **PlayerController** (std::shared_ptr< [Model::Player](#) > &entity)
- void [handleEvent](#) (const [KeyPressedEvent](#) &event) override
- void [handleEvent](#) (const [MoveEvent](#) &event) override
- void [handleEvent](#) (const [CollisionEvent](#) &event) override

Additional Inherited Members

7.35.1 Member Function Documentation

7.35.1.1 handleEvent() [1/3]

```
void PlayerController::handleEvent (
    const CollisionEvent & event ) [override], [virtual]
```

Reimplemented from [Controller::IController](#).

7.35.1.2 `handleEvent()` [2/3]

```
void PlayerController::handleEvent (
    const KeyPressedEvent & event ) [override], [virtual]
```

Reimplemented from [Controller::IController](#).

7.35.1.3 `handleEvent()` [3/3]

```
void PlayerController::handleEvent (
    const MoveEvent & event ) [override], [virtual]
```

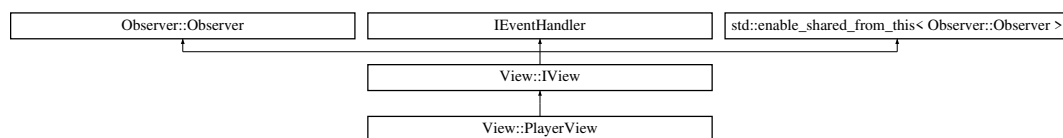
Reimplemented from [Controller::IController](#).

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

- `/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/Player↔Controller.h`
- `/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/controller/Player↔Controller.cpp`

7.36 `View::PlayerView` Class Reference

Inheritance diagram for `View::PlayerView`:



Public Member Functions

- **PlayerView** (const std::shared_ptr< [Model::Entity](#) > &entity, const std::shared_ptr< sf::RenderWindow > &window)

Additional Inherited Members

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

- `/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/Player↔View.h`

7.37 Utils::Random Class Reference

Class for [Random](#).

```
#include <Random.h>
```

Public Member Functions

- `~Random()`=default
Default destructor.
- `Random(const Random &)=delete`
Deleted copy constructor.
- `Random & operator= (const Random &)=delete`
Deleted assignment operator.
- `float random(float a, float b)`
Get random float in given interval.

Static Public Member Functions

- `static Random & getInstance()`
Get instance of [Random](#).

7.37.1 Detailed Description

Class for [Random](#).

7.37.2 Member Function Documentation

7.37.2.1 getInstance()

```
Random & Random::getInstance ( ) [static]
```

Get instance of [Random](#).

Returns

[Random](#)

7.37.2.2 operator=()

```
Random & Utils::Random::operator= (
    const Random & ) [delete]
```

Deleted assignment operator.

Returns

Random

7.37.2.3 random()

```
float Random::random (
    float a,
    float b )
```

Get random float in given interval.

Parameters

<i>a</i>	float - begin interval
<i>b</i>	float - end interval

Returns

float

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/Random.↵h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/util/Random.cpp

7.38 Utils::Resourceholder< Type > Struct Template Reference

Public Member Functions

- **Resourceholder** (std::string path)
- void **insert** (Model::Type type, const std::string &subPath)
- std::shared_ptr< Type > & **get** (Model::Type type)

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/util/Resourcemanager.↵h

7.39 Utils::Resourcemanager Class Reference

Public Member Functions

- **Resourcemanager** (const [Resourcemanager](#) &)=delete
- [Resourcemanager](#) & **operator=** (const [Resourcemanager](#) &)=delete
- void **addTexture** ([Model::Type](#) type, const std::string &subPath)
- void **addFont** ([Model::Type](#) type, const std::string &subPath)
- void **addSound** ([Model::Type](#) type, const std::string &subPath)
- const std::shared_ptr< [Resourceholder](#)< sf::Texture > > & **getTextures** () const
- const std::shared_ptr< [Resourceholder](#)< sf::Font > > & **getFonts** () const
- const std::shared_ptr< [Resourceholder](#)< sf::SoundBuffer > > & **getSounds** () const

Static Public Member Functions

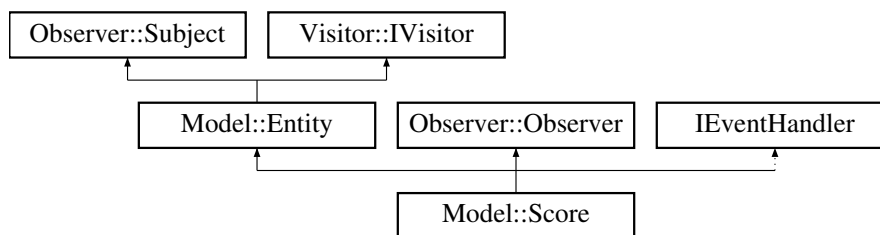
- static [Resourcemanager](#) & **getInstance** ()

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/util/Resourcemanager.h

7.40 Model::Score Class Reference

Inheritance diagram for Model::Score:



Public Member Functions

- [Model::Type](#) **getType** () const override
Get type of [Entity](#) object.
- void **setScore** (unsigned int score)
- void **onTrigger** (EventType type, const std::shared_ptr< [Event](#) > &event) override
Perform operation on trigger from Subject.
- void **handleEvent** (const [NewMaxHeightEvent](#) &event) override
- void **handleEvent** (const [CollisionEvent](#) &event) override
- void **move** (bool collision) override
move [Entity](#) object
- unsigned int **getScore** () const override

Additional Inherited Members

7.40.1 Member Function Documentation

7.40.1.1 `getScore()`

```
unsigned int Model::Score::getScore ( ) const [inline], [override], [virtual]
```

Reimplemented from [Model::Entity](#).

7.40.1.2 `getType()`

```
Model::Type Model::Score::getType ( ) const [inline], [override], [virtual]
```

Get type of [Entity](#) object.

Returns

[Model::Type](#)

Implements [Model::Entity](#).

7.40.1.3 `handleEvent()` [1/2]

```
void Score::handleEvent (
    const CollisionEvent & event ) [override], [virtual]
```

Reimplemented from [IEventHandler](#).

7.40.1.4 `handleEvent()` [2/2]

```
void Score::handleEvent (
    const NewMaxHeightEvent & event ) [override], [virtual]
```

Reimplemented from [IEventHandler](#).

7.40.1.5 move()

```
void Model::Score::move (
    bool collision ) [inline], [override], [virtual]
```

move [Entity](#) object

Implements [Model::Entity](#).

7.40.1.6 onTrigger()

```
void Model::Score::onTrigger (
    EventType type,
    const std::shared_ptr< Event > & event ) [inline], [override], [virtual]
```

Perform operation on trigger from Subject.

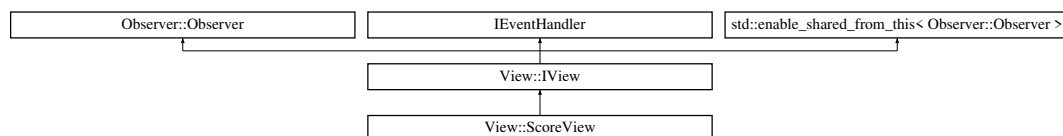
Implements [Observer::Observer](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Score.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/Score.cpp

7.41 View::ScoreView Class Reference

Inheritance diagram for View::ScoreView:



Public Member Functions

- **ScoreView** (std::shared_ptr< [Model::Score](#) > &entity, const std::shared_ptr< sf::RenderWindow > &window)

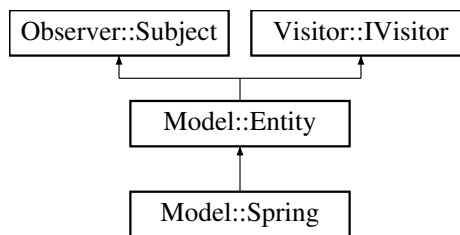
Additional Inherited Members

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/Score↔View.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/src/view/Score↔View.cpp

7.42 Model::Spring Class Reference

Inheritance diagram for Model::Spring:



Public Member Functions

- [Model::Type](#) `getType ()` const override
Get type of [Entity](#) object.
- void `move` (bool collision) override
move [Entity](#) object
- void `initBounds ()`
- void `visit` ([Model::Player](#) &player) override
- bool `isBonus ()` const override
- bool `isRemovable ()` const override

Additional Inherited Members

7.42.1 Member Function Documentation

7.42.1.1 `getType()`

```
Model::Type Model::Spring::getType ( ) const [override], [virtual]
```

Get type of [Entity](#) object.

Returns

[Model::Type](#)

Implements [Model::Entity](#).

7.42.1.2 `isBonus()`

```
bool Model::Spring::isBonus ( ) const [inline], [override], [virtual]
```

Reimplemented from [Model::Entity](#).

7.42.1.3 isRemovable()

```
bool Model::Spring::isRemovable ( ) const [inline], [override], [virtual]
```

Reimplemented from [Model::Entity](#).

7.42.1.4 move()

```
void Spring::move (
    bool collision ) [override], [virtual]
```

move [Entity](#) object

Implements [Model::Entity](#).

7.42.1.5 visit()

```
void Model::Spring::visit (
    Model::Player & player ) [inline], [override], [virtual]
```

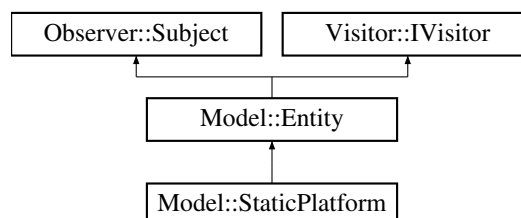
Reimplemented from [Model::Entity](#).

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

- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Spring.h](#)
- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/model/Spring.cpp](#)

7.43 Model::StaticPlatform Class Reference

Inheritance diagram for Model::StaticPlatform:



Public Member Functions

- [Model::Type](#) `getType ()` const override
Get type of [Entity](#) object.
- void [move](#) (bool collision) override
move [Entity](#) object

Additional Inherited Members

7.43.1 Member Function Documentation

7.43.1.1 `getType()`

```
Model::Type StaticPlatform::getType ( ) const [override], [virtual]
```

Get type of [Entity](#) object.

Returns

[Model::Type](#)

Implements [Model::Entity](#).

7.43.1.2 `move()`

```
void StaticPlatform::move (
    bool collision ) [override], [virtual]
```

move [Entity](#) object

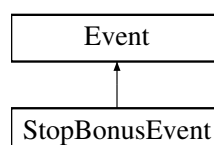
Implements [Model::Entity](#).

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

- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/StaticPlatform.h](#)
- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/model/StaticPlatform.cpp](#)

7.44 StopBonusEvent Class Reference

Inheritance diagram for StopBonusEvent:



Public Member Functions

- **StopBonusEvent** (std::shared_ptr< [Model::Entity](#) > bonus)
- void [send](#) ([IEventHandler](#) &handler) const override
- const std::shared_ptr< [Model::Entity](#) > & **getBonus** () const

Additional Inherited Members

7.44.1 Member Function Documentation

7.44.1.1 [send\(\)](#)

```
void StopBonusEvent::send (
    IEventHandler & handler ) const [inline], [override], [virtual]
```

Reimplemented from [Event](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h

7.45 Utils::Stopwatch Class Reference

Class for [Stopwatch](#).

```
#include <Stopwatch.h>
```

Public Member Functions

- **Stopwatch** ()=default
Private default constructor.
- **~Stopwatch** ()=default
Default destructor.
- **Stopwatch** (const [Stopwatch](#) &)=delete
Deleted copy constructor.
- [Stopwatch](#) & **operator=** (const [Stopwatch](#) &)=delete
Deleted assignment operator.
- void **start** ()
Start [Stopwatch](#).
- float [lap](#) ()
Lap one round and return found delta.
- float [getDelta](#) () const
Get latest delta.
- void **addTimer** (unsigned int key, float amount)
- bool **checkTimer** (unsigned int key)

Static Public Member Functions

- static [Stopwatch](#) & [getInstance](#) ()
Get instance of [Stopwatch](#).

Public Attributes

- std::shared_ptr< [Model::Entity](#) > mPlayer

7.45.1 Detailed Description

Class for [Stopwatch](#).

7.45.2 Member Function Documentation

7.45.2.1 [getDelta\(\)](#)

```
float Stopwatch::getDelta ( ) const
```

Get latest delta.

Returns

float

7.45.2.2 [getInstance\(\)](#)

```
Stopwatch & Stopwatch::getInstance ( ) [static]
```

Get instance of [Stopwatch](#).

Returns

[Stopwatch](#)

7.45.2.3 lap()

```
float Stopwatch::lap ( )
```

Lap one round and return found delta.

Returns

float

7.45.2.4 operator=()

```
Stopwatch & Utils::Stopwatch::operator= (
    const Stopwatch & ) [delete]
```

Deleted assignment operator.

Returns

[Stopwatch](#)

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

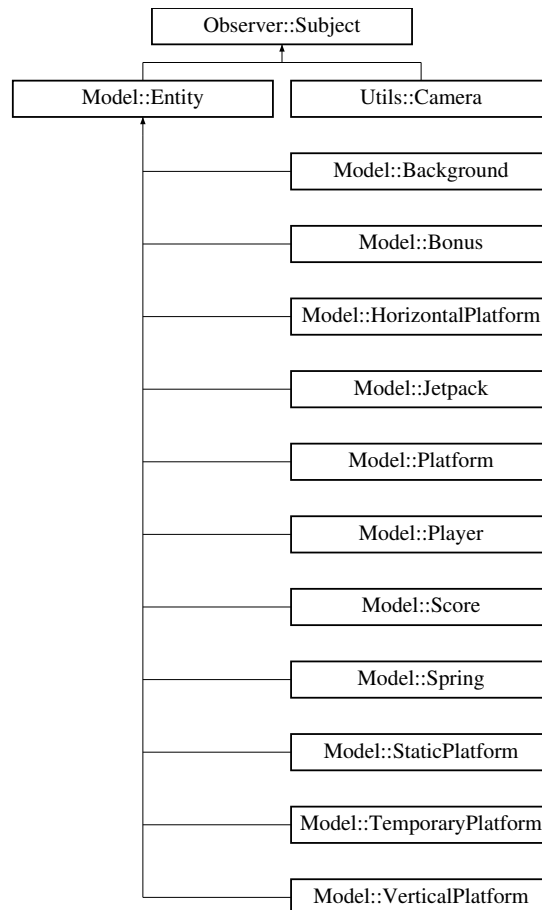
- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/Stopwatch.h](#)↔
- [/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/util/Stopwatch.cpp](#)↔

7.46 Observer::Subject Class Reference

Class for [Subject](#) / Observable of observer pattern.

```
#include <Subject.h>
```

Inheritance diagram for Observer::Subject:



Public Member Functions

- **Subject** ()=default
Default constructor.
- virtual **~Subject** ()=default
Default destructor.
- void **add** (const std::shared_ptr< [Observer](#) > &observer)
Register [Observer](#) to mObservers.
- const std::vector< std::shared_ptr< [Observer](#) > > & **getObservers** () const
- void **clear** ()
Clear all Observers from mObservers.
- void **trigger** (EventType type, const std::shared_ptr< [Event](#) > &event) const
Trigger registered Observers.

7.46.1 Detailed Description

Class for [Subject](#) / Observable of observer pattern.

7.46.2 Member Function Documentation

7.46.2.1 add()

```
void Observer::Subject::add (
    const std::shared_ptr< Observer > & observer ) [inline]
```

Register [Observer](#) to mObservers.

Parameters

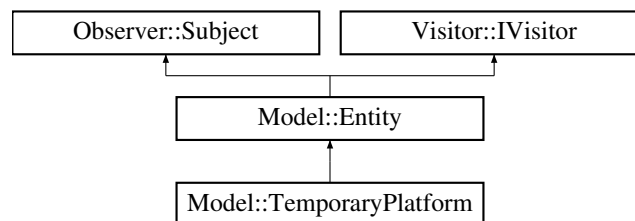
<i>observer</i>	Observer to be added
-----------------	--------------------------------------

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Subject.h

7.47 Model::TemporaryPlatform Class Reference

Inheritance diagram for Model::TemporaryPlatform:



Public Member Functions

- [Model::Type](#) [getType](#) () const override
Get type of [Entity](#) object.
- void [move](#) (bool collision) override
move [Entity](#) object

Additional Inherited Members

7.47.1 Member Function Documentation

7.47.1.1 getType()

```
Model::Type TemporaryPlatform::getType ( ) const [override], [virtual]
```

Get type of [Entity](#) object.

Returns

[Model::Type](#)

Implements [Model::Entity](#).

7.47.1.2 move()

```
void TemporaryPlatform::move (
    bool collision ) [override], [virtual]
```

move [Entity](#) object

Implements [Model::Entity](#).

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/TemporaryPlatform.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/model/TemporaryPlatform.cpp

7.48 Utils::Utilities Class Reference

Class for [Utilities](#).

```
#include <Utilities.h>
```

Static Public Member Functions

- static bool [checkCollision](#) (const std::shared_ptr< [Model::Entity](#) > &l, const std::shared_ptr< [Model::Entity](#) > &r)
Check if there is a collision between two Entities.
- static bool [checkWeight](#) (float &rand, float weight)
Check random spawn rate of total chance.

7.48.1 Detailed Description

Class for [Utilities](#).

7.48.2 Member Function Documentation

7.48.2.1 checkCollision()

```
bool Utilities::checkCollision (
    const std::shared_ptr< Model::Entity > & l,
    const std::shared_ptr< Model::Entity > & r ) [static]
```

Check if there is a collision between two Entities.

Parameters

<i>l</i>	first entity
<i>r</i>	second entity

Returns

bool

7.48.2.2 checkWeight()

```
bool Utilities::checkWeight (
    float & rand,
    float weight ) [static]
```

Check random spawn rate of total chance.

Parameters

<i>rand</i>	float - random spawn rate
<i>weight</i>	float - total chance

Returns

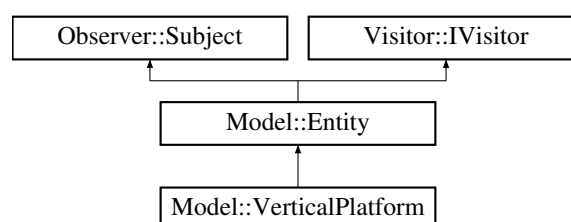
true if rand <= weight

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/Utilities.h↔
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/util/Utilities.cpp

7.49 Model::VerticalPlatform Class Reference

Inheritance diagram for Model::VerticalPlatform:



Public Member Functions

- [Model::Type](#) `getType ()` const override
Get type of [Entity](#) object.
- void [move](#) (bool collision) override
move [Entity](#) object
- void `initBounds ()`

Additional Inherited Members

7.49.1 Member Function Documentation

7.49.1.1 `getType()`

```
Model::Type VerticalPlatform::getType ( ) const [override], [virtual]
```

Get type of [Entity](#) object.

Returns

[Model::Type](#)

Implements [Model::Entity](#).

7.49.1.2 `move()`

```
void VerticalPlatform::move (
    bool collision ) [override], [virtual]
```

move [Entity](#) object

Implements [Model::Entity](#).

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

- `/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/VerticalPlatform.h`
- `/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/model/VerticalPlatform.cpp`

7.50 World Class Reference

Class for [World](#), holds all the entities and is used for the overall game logic.

```
#include <World.h>
```

Public Member Functions

- **World** (std::shared_ptr< [Model::AbstractFactory](#) > &factory)
- void **initWorld** ()
Initialize starting-world.
- void **events** (const std::string &move, bool isPressed) const
Process events such as player input.
- void **update** ()
Gets called every tick from [Game](#) loop, updates ever [Model](#).
- void **render** () const
Render all entities on screen.
- void **generateEntity** ()
Generate new Entity.
- void **spawnPlatform** (float x, float y)
Spawn random Platform object into [World](#).
- void **spawnBonus** (float x, float y)
Spawn random Bonus object AND Platform object into [World](#).
- void **spawnEntity** (float x, float y, [Model::Type](#) type)
Spawn provided Entity into [World](#).
- bool **checkDifficulty** ()
- void **addEntity** (const std::shared_ptr< [Model::Entity](#) > &entity)
Add entity.
- void **removeEntities** ()
Remove unused Entities or those that are out of view.
- void **destroy** ()
- const std::shared_ptr< [Model::Score](#) > &**getScore** () const
- bool **isPlaying** () const

7.50.1 Detailed Description

Class for [World](#), holds all the entities and is used for the overall game logic.

7.50.2 Member Function Documentation

7.50.2.1 addEntity()

```
void World::addEntity (
    const std::shared_ptr< Model::Entity > & entity )
```

Add entity.

Parameters

<i>entity</i>	Entity to be added to mEntities
---------------	---------------------------------

7.50.2.2 events()

```
void World::events (
    const std::string & move,
    bool isPressed ) const
```

Process events such as player input.

Parameters

<i>move</i>	std::string - which key is pressed
<i>isPressed</i>	bool - is key pressed

7.50.2.3 spawnBonus()

```
void World::spawnBonus (
    float x,
    float y )
```

Spawn random Bonus object AND Platform object into [World](#).

Parameters

<i>x</i>	float
<i>y</i>	float

7.50.2.4 spawnEntity()

```
void World::spawnEntity (
    float x,
    float y,
    Model::Type type )
```

Spawn provided Entity into [World](#).

Parameters

<i>x</i>	float
<i>y</i>	float
<i>type</i>	Model::Type - Sort of Entity to be spawned

7.50.2.5 spawnPlatform()

```
void World::spawnPlatform (
    float x,
    float y )
```

Spawn random Platform object into [World](#).

Parameters

<i>x</i>	float
<i>y</i>	float

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

- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/World.h
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/src/World.cpp

Chapter 8

File Documentation

8.1 AbstractFactory.h

```
1 //
2 // Created by Pablo Deputter on 21/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_ABSTRACTFACTORY_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_ABSTRACTFACTORY_H
7
8 #include "Score.h"
9 #include "controller/IController.h"
10 #include "model/Entity.h"
11
12 namespace Model {
13 // Used by world to create new entities without knowing anything sfml-related (VIEW)
14
15 // The game class provides pointer to concrete factory to world, so it can create
16 // entities that have correct view attached
17 class AbstractFactory
18 {
19 public:
20     AbstractFactory() = default;
21
22     virtual ~AbstractFactory() = default;
23
24     virtual std::shared_ptr<Model::Player> createPlayer() = 0;
25
26     virtual std::shared_ptr<Model::Entity> createStaticPlatform() = 0;
27
28     virtual std::shared_ptr<Model::Entity> createHorizontalPlatform() = 0;
29
30     virtual std::shared_ptr<Model::Entity> createVerticalPlatform() = 0;
31
32     virtual std::shared_ptr<Model::Entity> createTemporaryPlatform() = 0;
33
34     virtual std::shared_ptr<Model::Entity> createSpring() = 0;
35
36     virtual std::shared_ptr<Model::Entity> createJetpack() = 0;
37
38     virtual std::shared_ptr<Model::Entity> createBackground() = 0;
39
40     virtual std::shared_ptr<Model::Score> createScore() = 0;
41 };
42 } // namespace Model
43
44 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_ABSTRACTFACTORY_H
```

8.2 BonusController.h

```
1 //
2 // Created by Pablo Deputter on 07/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_BONUSCONTROLLER_H
6 #define DOODLEJUMP_BONUSCONTROLLER_H
7
```

```

8 #include "IController.h"
9
10 namespace Controller {
11
12 class BonusController : public IController
13 {
14 public:
15     BonusController(std::shared_ptr<Model::Entity>& entity) : IController(entity) {}
16
17     BonusController() = default;
18
19     ~BonusController() override = default;
20 };
21 } // namespace Controller
22
23 #endif // DOODLEJUMP_BONUSCONTROLLER_H

```

8.3 IController.h

```

1 //
2 // Created by Pablo Deputter on 19/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_ICONTROLLER_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_ICONTROLLER_H
7
8 #include "model/Entity.h"
9 #include "model/Player.h"
10
11 #include <memory>
12
13 namespace Controller {
14
15 class IController : public Observer::Observer, public IEventHandler
16 {
17 protected:
18     std::shared_ptr<Model::Entity> mEntity;
19
20 public:
21     IController() = default;
22
23     IController(std::shared_ptr<Model::Entity>& entity) : mEntity(entity) {}
24
25     virtual ~IController() override = default;
26
27     const std::shared_ptr<Model::Entity>& getEntity() const { return mEntity; }
28
29     void onTrigger(EventType type, const std::shared_ptr<Event>& event) override {
30         event->send(*this); }
31
32     void handleEvent(const KeyPressedEvent& event) override {}
33
34     void handleEvent(const MoveEvent& event) override;
35
36     void handleEvent(const CollisionEvent& event) override {}
37 };
38 } // namespace Controller
39 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_ICONTROLLER_H

```

8.4 PlatformController.h

```

1 //
2 // Created by Pablo Deputter on 20/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_PLATFORMCONTROLLER_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_PLATFORMCONTROLLER_H
7
8 #include "IController.h"
9
10 namespace Controller {
11
12 class PlatformController : public IController
13 {
14 public:
15     PlatformController(std::shared_ptr<Model::Entity>& entity) : IController(entity) {}
16
17     PlatformController() = default;

```



```

18
19     ~PlatformController() override = default;
20 };
21 } // namespace Controller
22
23 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_PLATFORMCONTROLLER_H

```

8.5 PlayerController.h

```

1 //
2 // Created by Pablo Deputter on 19/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_PLAYERCONTROLLER_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_PLAYERCONTROLLER_H
7
8 #include "IController.h"
9
10 namespace Controller {
11
12     class PlayerController : public IController
13     {
14     public:
15         explicit PlayerController(std::shared_ptr<Model::Entity>& entity) : IController(entity) {}
16
17         explicit PlayerController(std::shared_ptr<Model::Player>& entity) { mEntity = entity; }
18
19         PlayerController() = default;
20
21         ~PlayerController() override = default;
22
23         void handleEvent(const KeyPressedEvent& event) override;
24
25         void handleEvent(const MoveEvent& event) override;
26
27         void handleEvent(const CollisionEvent& event) override;
28     };
29 } // namespace Controller
30
31 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_PLAYERCONTROLLER_H

```

8.6 Event.h

```

1 //
2 // Created by Pablo Deputter on 08/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_EVENT_H
6 #define DOODLEJUMP_EVENT_H
7
8 #include "Settings.h"
9
10 #include <iostream>
11 #include <memory>
12 #include <string>
13 #include <utility>
14
15 namespace Model {
16     class Entity;
17     class Player;
18 } // namespace Model
19
20 enum class EventType
21 {
22     OUT_OF_VIEW = 0,
23     DRAW,
24     NEW_MAX_HEIGHT,
25     COLLISION,
26     KEY_PRESSED,
27     MOVE,
28     STOP_BONUS,
29     NEW_DIFFICULTY
30 };
31
32 class Event;
33 class DrawEvent;
34 class OutOfViewEvent;
35 class NewMaxHeightEvent;
36 class CollisionEvent;

```

```

37 class KeyPressedEvent;
38 class MoveEvent;
39 class StopBonusEvent;
40 class NewDifficultyEvent;
41
42 class IEventHandler
43 {
44 public:
45     virtual void handleEvent(const Event& event) {}
46
47     virtual void handleEvent(const DrawEvent& event) {}
48
49     virtual void handleEvent(const OutOfViewEvent& event) {}
50
51     virtual void handleEvent(const NewMaxHeightEvent& event) {}
52
53     virtual void handleEvent(const CollisionEvent& event) {}
54
55     virtual void handleEvent(const KeyPressedEvent& event) {}
56
57     virtual void handleEvent(const MoveEvent& event) {}
58
59     virtual void handleEvent(const StopBonusEvent& event) {}
60
61     virtual void handleEvent(const NewDifficultyEvent& event) {}
62 };
63
64 class Event
65 {
66 protected:
67     EventType mEvent;
68
69 public:
70     explicit Event(EventType mEvent) : mEvent(mEvent) {}
71
72     Event() = default;
73
74     virtual ~Event() = default;
75
76     virtual void send(IEventHandler& handler) const { handler.handleEvent(*this); }
77
78     [[nodiscard]] EventType getEvent() const { return mEvent; }
79 };
80
81 class DrawEvent : public Event
82 {
83 public:
84     DrawEvent() : Event() {}
85
86     ~DrawEvent() override = default;
87
88     void send(IEventHandler& handler) const override { handler.handleEvent(*this); }
89 };
90
91 class OutOfViewEvent : public Event
92 {
93 public:
94     OutOfViewEvent() : Event() {}
95
96     ~OutOfViewEvent() override = default;
97
98     void send(IEventHandler& handler) const override { handler.handleEvent(*this); }
99 };
100
101 class NewMaxHeightEvent : public Event
102 {
103 private:
104     float mLastHeight;
105     float mNewHeight;
106
107 public:
108     NewMaxHeightEvent(float lastHeight, float newHeight) : Event(), mLastHeight(lastHeight),
109         mNewHeight(newHeight)
110     {
111
112     }
113
114     ~NewMaxHeightEvent() override = default;
115
116     void send(IEventHandler& handler) const override { handler.handleEvent(*this); }
117
118     float getLastHeight() const { return mLastHeight; }
119
120     float getNewHeight() const { return mNewHeight; }
121 };
122
123 class CollisionEvent : public Event
124 {

```

```

123 private:
124     std::shared_ptr<Model::Entity> mEntity;
125     std::shared_ptr<Model::Player> mPlayer;
126
127 public:
128     explicit CollisionEvent(std::shared_ptr<Model::Entity> entity, std::shared_ptr<Model::Player>
        player)
129         : Event(), mEntity(std::move(entity)), mPlayer(std::move(player))
130     {
131     }
132
133     ~CollisionEvent() override = default;
134
135     void send(EventHandler& handler) const override { handler.handleEvent(*this); }
136
137     const std::shared_ptr<Model::Entity>& getEntity() const { return mEntity; }
138
139     const std::shared_ptr<Model::Player>& getPlayer() const { return mPlayer; }
140 };
141
142 class KeyPressedEvent : public Event
143 {
144 private:
145     std::string mKey;
146     bool mIsPressed;
147
148 public:
149     KeyPressedEvent(std::string key, bool isPressed) : Event(), mKey(std::move(key)),
        mIsPressed(isPressed) {}
150
151     ~KeyPressedEvent() override = default;
152
153     void send(EventHandler& handler) const override { handler.handleEvent(*this); }
154
155     const std::string& getKey() const { return mKey; }
156
157     bool isPressed() const { return mIsPressed; }
158 };
159
160 class MoveEvent : public Event
161 {
162 private:
163     bool mCollided;
164
165 public:
166     explicit MoveEvent(bool collided) : Event(), mCollided(collided) {}
167
168     ~MoveEvent() override = default;
169
170     void send(EventHandler& handler) const override { handler.handleEvent(*this); }
171
172     bool isCollided() const { return mCollided; }
173 };
174
175 class StopBonusEvent : public Event
176 {
177 private:
178     std::shared_ptr<Model::Entity> mBonus;
179
180 public:
181     explicit StopBonusEvent(std::shared_ptr<Model::Entity> bonus) : Event(),
        mBonus(std::move(bonus)) {}
182
183     ~StopBonusEvent() override = default;
184
185     void send(EventHandler& handler) const override { handler.handleEvent(*this); }
186
187     const std::shared_ptr<Model::Entity>& getBonus() const { return mBonus; }
188 };
189
190 class NewDifficultyEvent : public Event
191 {
192 private:
193     Settings::Difficulty mDifficulty;
194
195 public:
196     explicit NewDifficultyEvent(Settings::Difficulty difficulty) : Event(), mDifficulty(difficulty)
        {}
197
198     ~NewDifficultyEvent() override = default;
199
200     void send(EventHandler& handler) const override { handler.handleEvent(*this); }
201
202     [[nodiscard]] Settings::Difficulty getDifficulty() const { return mDifficulty; }
203 };
204
205 #endif // DOODLEJUMP_EVENT_H

```

8.7 HighScore.h

```

1 //
2 // Created by Pablo Deputter on 13/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_HIGHSORE_H
6 #define DOODLEJUMP_HIGHSORE_H
7
8 #include "util/Exception.h"
9
10 #include <fstream>
11 #include <iostream>
12 #include <memory>
13 #include <ostream>
14 #include <string>
15 #include <vector>
16
17 struct HighScoreScore
18 {
19     unsigned int mScore;
20     std::string mName;
21
22     HighScoreScore() : mScore(0), mName(std::string()) {}
23
24     HighScoreScore(unsigned int score, std::string name) : mScore(score), mName(std::move(name)) {}
25
26     virtual ~HighScoreScore() = default;
27
28     [[nodiscard]] std::string toString() const { return mName + " - " + std::to_string(mScore); }
29
30     friend std::ostream& operator<<(std::ostream& os, const HighScoreScore& score)
31     {
32         os << score.mName << " - " << score.mScore << "\n";
33         return os;
34     }
35 };
36
37 class HighScore
38 {
39 private:
40     std::string mPath;
41     unsigned int mQuantity;
42     std::vector<std::shared_ptr<HighScoreScore> mScores;
43
44     HighScore() = default;
45
46     HighScore(std::string path, unsigned int quantity) : mPath(path), mQuantity(quantity)
47     {
48         try {
49             load();
50         } catch (const std::exception& exc) {
51             std::cerr << exc.what();
52         }
53     }
54
55 public:
56     HighScore(const HighScore&) = delete;
57
58     HighScore& operator=(const HighScore&) = delete;
59
60     ~HighScore() { save(); }
61
62     static HighScore& getInstance();
63
64     void load();
65
66     void save();
67
68     void add(const std::shared_ptr<HighScoreScore>& score);
69
70     [[nodiscard]] const std::vector<std::shared_ptr<HighScoreScore>& getScores() const { return
mScores; }
71
72     [[nodiscard]] unsigned int getHighScore() const
73     {
74         if (mScores.empty()) {
75             return 0;
76         }
77         return mScores.front()->mScore;
78     }
79 };
80
81 #endif // DOODLEJUMP_HIGHSORE_H

```

8.8 IVisitor.h

```

1 //
2 // Created by Pablo Deputter on 07/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_IVISITOR_H
6 #define DOODLEJUMP_IVISITOR_H
7
8 #include <memory>
9
10 namespace Model {
11 class Player;
12 }
13
14 namespace Visitor {
15
16 class IVisitor
17 {
18 public:
19     virtual void visit(Model::Player& player) = 0;
20 };
21 } // namespace Visitor
22
23 #endif // DOODLEJUMP_IVISITOR_H

```

8.9 Background.h

```

1 //
2 // Created by Pablo Deputter on 29/11/2021.
3 //
4
5 #ifndef DOODLEJUMP_BACKGROUND_H
6 #define DOODLEJUMP_BACKGROUND_H
7
8 #include "Entity.h"
9
10 namespace Model {
11
12 class Background : public Entity
13 {
14 private:
15 public:
16     Background() = default;
17
18     ~Background() override = default;
19
20     Model::Type getType() const override { return Model::Type::eBackground; }
21
22     void move(bool collision) override {}
23 };
24 } // namespace Model
25
26 #endif // DOODLEJUMP_BACKGROUND_H

```

8.10 Bonus.h

```

1 //
2 // Created by Pablo Deputter on 16/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_BONUS_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_BONUS_H
7
8 #include "Entity.h"
9
10 namespace Model {
11
12 class Bonus : public Entity
13 {
14 public:
15     enum Sort
16     {
17         eJetpack = 0,
18         eSpring = 1,
19     };
20
21 private:
22     Bonus::Sort mSort;
23 };
24
25 #endif

```

```

26
27 public:
32     Model::Type getType() const override { return Model::Type::eBonus; }
37     Sort getMSort() const;
42     void setMSort(Sort sort);
43 };
44 } // namespace Model
45
46 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_BONUS_H

```

8.11 Entity.h

```

1 //
2 // Created by Pablo Deputter on 13/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_ENTITY_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_ENTITY_H
7
8 #include "Subject.h"
9 #include "util/Camera.h"
10 #include "util/Stopwatch.h"
11
12 #include "IVisitor.h"
13
14 #include <iostream>
15 #include <string>
16
17 namespace Model {
18     enum Type
19     {
20         ePlayer = 0,
21         eBonus = 1,
22         eStatic = 2,
23         eHorizontal = 3,
24         eVertical = 4,
25         eTemporary = 5,
26         eBackground = 6,
27         eJetpack = 7,
28         eSpring = 8,
29         eScore = 9
30     };
31
32     class CollisionBox
33     {
34     private:
35         float mLeft;
36         float mWidth;
37         float mBottom;
38         float mHeight;
39
40     public:
41         CollisionBox() = default;
42
43         CollisionBox(float left, float width, float bottom, float height)
44             : mLeft(left), mWidth(width), mBottom(bottom), mHeight(height)
45         {
46         }
47
48         ~CollisionBox() = default;
49
50         [[nodiscard]] float getLeft() const { return mLeft; }
51
52         [[nodiscard]] float getWidth() const { return mWidth; }
53
54         [[nodiscard]] float getBottom() const { return mBottom; }
55
56         [[nodiscard]] float getHeight() const { return mHeight; }
57     };
58
59     class Entity : public Observer::Subject, public Visitor::IVisitor
60     {
61     protected:
62         float mX;
63         float mY;
64         //
65         //
66         float mWidth{};
67         float mHeight{};
68         bool mRemoveFlag;
69         unsigned int mScore;
70
71         float mSpawnRate;

```

```

81
82 public:
83     Entity() : mX(0.f), mY(0.f), mWidth(0.f), mHeight(0.f), mRemoveFlag(false), mScore(0),
84             mSpawnRate(0.f) {}
85
86     Entity(unsigned int score, float spawnRate)
87         : mX(0.f), mY(0.f), mWidth(0.f), mHeight(0.f), mRemoveFlag(false), mScore(score),
88         mSpawnRate(spawnRate)
89     {
90     }
91
92     virtual ~Entity() = default;
93     float getX() const;
94     float getY() const;
95     void setX(float x);
96     void setY(float y);
97
98     float getWidth() const;
99     float getHeight() const;
100    void setWidth(float width);
101    void setHeight(float height);
102    virtual void move(bool collision) = 0;
103    void move(float x, float y);
104    virtual Model::Type getType() const = 0;
105    virtual void onDestroy();
106
107    void visit(Model::Player& player) override { std::cout << "Entity visit\n"; }
108
109    virtual void accept(const std::shared_ptr<Visitor::IVisitor>& visitor) {}
110
111    // TODO - jetpack
112    void setRemoveFlag(bool flag)
113    {
114        mRemoveFlag = flag;
115        onDestroy();
116    }
117
118    virtual bool getRemovable() const { return mRemoveFlag; }
119
120    virtual bool isRemovable() const { return mRemoveFlag; }
121
122    virtual bool isBonus() const { return false; }
123
124    void setScore(unsigned int score) { mScore = score; }
125
126    virtual unsigned int getScore() const { return mScore; }
127
128    virtual float getSpawnRate() const { return mSpawnRate; }
129
130    void setSpawnRate(float spawnRate) { mSpawnRate = spawnRate; }
131 };
132 } // namespace Model
133
134 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_ENTITY_H

```

8.12 HorizontalPlatform.h

```

1 //
2 // Created by Pablo Deputter on 05/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_HORIZONTALPLATFORM_H
6 #define DOODLEJUMP_HORIZONTALPLATFORM_H
7
8 #include "Entity.h"
9 #include "util/Random.h"
10
11 namespace Model {
12
13     class HorizontalPlatform : public Entity
14     {
15     private:
16         std::pair<float, float> mBounds;
17         bool mMovingForward;
18         bool mInit;
19
20     public:
21         HorizontalPlatform() : Entity(5, .10f), mBounds({0.f, 0.f}), mMovingForward(false), mInit(false)
22         {}
23
24         ~HorizontalPlatform() override = default;
25

```

```

25         Model::Type getType() const override;
26
27         void move(bool collision) override;
28
29         void initBounds();
30 };
31 } // namespace Model
32
33 #endif // DOODLEJUMP_HORIZONTALPLATFORM_H

```

8.13 Jetpack.h

```

1 //
2 // Created by Pablo Deputter on 06/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_JETPACK_H
6 #define DOODLEJUMP_JETPACK_H
7
8 #include "Player.h"
9
10 namespace Model {
11
12     class Jetpack : public Entity
13     {
14     private:
15         std::pair<float, float> mBounds;
16         bool mMovingDown;
17         bool mInit;
18
19     public:
20         bool mStarted;
21
22     public:
23         Jetpack(bool started) : Entity(20, .25f), mMovingDown(false), mInit(false), mStarted(started) {}
24
25         ~Jetpack() override = default;
26
27         [[nodiscard]] Model::Type getType() const override;
28
29         void move(bool collision) override;
30
31         void initBounds();
32
33         // TODO - jetpack
34         void visit(Model::Player& player) override
35         {
36             std::cout << "visit\n";
37             if (mStarted) {
38                 std::cout << "return player state\n";
39                 player.setDrag(0.006f);
40                 return;
41             }
42             Utils::Stopwatch::getInstance().addTimer(Model::eJetpack, 2.5f);
43             player.setDrag(0.f);
44             mStarted = true;
45         }
46         // TODO - observer pattern mss
47         // event met player dus dan kan die accepten als gedaan is
48         bool isRemovable() const override
49         {
50             // moet removeflag en gestart zijn en de timer moet afgelopen zijn
51             if (mStarted && mStarted && Utils::Stopwatch::getInstance().checkTimer(Model::eJetpack))
52             {
53                 std::shared_ptr<Model::Entity> jetpack = std::make_shared<Model::Jetpack>(true);
54
55                 trigger(EventType::STOP_BONUS, std::make_shared<StopBonusEvent>(jetpack));
56                 std::cout << "jetpackStopped\n";
57                 return true;
58             }
59             return false;
60         }
61         bool isBonus() const override { return true; }
62     };
63 } // namespace Model
64
65 #endif // DOODLEJUMP_JETPACK_H

```


8.14 Platform.h

```

1 //
2 // Created by Pablo Deputter on 13/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_PLATFORM_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_PLATFORM_H
7
8 #include "Entity.h"
9 #include "util/Random.h"
10 #include <iostream>
11
12 namespace Model {
13
14 class Platform : public Entity
15 {
16 private:
17     Model::Type mSort;
18
19     std::pair<float, float> mBoundX = {0.f, 0.f};
20     bool mMovingForward = true;
21
22     std::pair<float, float> mBoundY = {0.f, 0.f};
23     bool mMovingDown = true;
24
25 public:
26     explicit Platform(Model::Type sort) : mSort(sort) {}
27
28     Platform()
29     {
30         float rand = Utils::Random::getInstance().random(0.f, 1.f);
31         // TODO - remove
32         // std::cout << rand << std::endl;
33         if (rand <= .75f)
34             mSort = eStatic;
35         else if (.85f >= rand >= .75f)
36             mSort = eHorizontal;
37         else if (.95f >= rand >= .85f)
38             mSort = eVertical;
39         else
40             mSort = eTemporary;
41     }
42
43     ~Platform() override = default;
44
45     Model::Type getType() const override { return mSort; }
46
47     void move(bool collision) override;
48
49     void initBounds();
50 };
51 } // namespace Model
52
53 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_PLATFORM_H

```

8.15 Player.h

```

1 //
2 // Created by Pablo Deputter on 14/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_PLAYER_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_PLAYER_H
7
8 #include "Entity.h"
9 #include "IVisitor.h"
10 #include <iostream>
11
12 namespace Model {
13
14 class Player : public Entity, public Observer::Observer, public IEventHandler
15 {
16 private:
17     std::pair<float, float> mVelocity;
18     std::pair<float, float> mDirection;
19     const float mMaxVelocity = 0.20f;
20     const float mAcceleration = 0.015f;
21     float mDrag = 0.005f;
22     bool mIsMovingLeft;
23     bool mIsMovingRight;
24
25 public:
26     Player() : mVelocity({0.f, 0.f}), mDirection({0.f, 0.f}), mIsMovingLeft(false),
27         mIsMovingRight(false) {}
28
29     void move() override;
30
31     void initBounds() override;
32
33     void initVelocity() override;
34
35     void initDirection() override;
36
37     void initAcceleration() override;
38
39     void initDrag() override;
40
41     void initIsMovingLeft() override;
42
43     void initIsMovingRight() override;
44
45     void update() override;
46
47     void handleEvent(IEvent* event) override;
48
49     void observe() override;
50
51     void unobserve() override;
52
53     void reset() override;
54
55     void resetVelocity() override;
56
57     void resetDirection() override;
58
59     void resetAcceleration() override;
60
61     void resetDrag() override;
62
63     void resetIsMovingLeft() override;
64
65     void resetIsMovingRight() override;
66
67     void resetUpdate() override;
68
69     void resetHandleEvent() override;
70
71     void resetObserve() override;
72
73     void resetUnobserve() override;
74
75     void resetReset() override;
76
77     void resetResetVelocity() override;
78
79     void resetResetDirection() override;
80
81     void resetResetAcceleration() override;
82
83     void resetResetDrag() override;
84
85     void resetResetIsMovingLeft() override;
86
87     void resetResetIsMovingRight() override;
88
89     void resetResetUpdate() override;
90
91     void resetResetHandleEvent() override;
92
93     void resetResetObserve() override;
94
95     void resetResetUnobserve() override;
96
97     void resetResetReset() override;
98
99     void resetResetResetVelocity() override;
100
101     void resetResetResetDirection() override;
102
103     void resetResetResetAcceleration() override;
104
105     void resetResetResetDrag() override;
106
107     void resetResetResetIsMovingLeft() override;
108
109     void resetResetResetIsMovingRight() override;
110
111     void resetResetResetUpdate() override;
112
113     void resetResetResetHandleEvent() override;
114
115     void resetResetResetObserve() override;
116
117     void resetResetResetUnobserve() override;
118
119     void resetResetResetReset() override;
120
121     void resetResetResetResetVelocity() override;
122
123     void resetResetResetResetDirection() override;
124
125     void resetResetResetResetAcceleration() override;
126
127     void resetResetResetResetDrag() override;
128
129     void resetResetResetResetIsMovingLeft() override;
130
131     void resetResetResetResetIsMovingRight() override;
132
133     void resetResetResetResetUpdate() override;
134
135     void resetResetResetResetHandleEvent() override;
136
137     void resetResetResetResetObserve() override;
138
139     void resetResetResetResetUnobserve() override;
140
141     void resetResetResetResetReset() override;
142
143     void resetResetResetResetResetVelocity() override;
144
145     void resetResetResetResetResetDirection() override;
146
147     void resetResetResetResetResetAcceleration() override;
148
149     void resetResetResetResetResetDrag() override;
150
151     void resetResetResetResetResetIsMovingLeft() override;
152
153     void resetResetResetResetResetIsMovingRight() override;
154
155     void resetResetResetResetResetUpdate() override;
156
157     void resetResetResetResetResetHandleEvent() override;
158
159     void resetResetResetResetResetObserve() override;
160
161     void resetResetResetResetResetUnobserve() override;
162
163     void resetResetResetResetResetReset() override;
164
165     void resetResetResetResetResetResetVelocity() override;
166
167     void resetResetResetResetResetResetDirection() override;
168
169     void resetResetResetResetResetResetAcceleration() override;
170
171     void resetResetResetResetResetResetDrag() override;
172
173     void resetResetResetResetResetResetIsMovingLeft() override;
174
175     void resetResetResetResetResetResetIsMovingRight() override;
176
177     void resetResetResetResetResetResetUpdate() override;
178
179     void resetResetResetResetResetResetHandleEvent() override;
180
181     void resetResetResetResetResetResetObserve() override;
182
183     void resetResetResetResetResetResetUnobserve() override;
184
185     void resetResetResetResetResetResetReset() override;
186
187     void resetResetResetResetResetResetResetVelocity() override;
188
189     void resetResetResetResetResetResetResetDirection() override;
190
191     void resetResetResetResetResetResetResetAcceleration() override;
192
193     void resetResetResetResetResetResetResetDrag() override;
194
195     void resetResetResetResetResetResetResetIsMovingLeft() override;
196
197     void resetResetResetResetResetResetResetIsMovingRight() override;
198
199     void resetResetResetResetResetResetResetUpdate() override;
200
201     void resetResetResetResetResetResetResetHandleEvent() override;
202
203     void resetResetResetResetResetResetResetObserve() override;
204
205     void resetResetResetResetResetResetResetUnobserve() override;
206
207     void resetResetResetResetResetResetResetReset() override;
208
209     void resetResetResetResetResetResetResetResetVelocity() override;
210
211     void resetResetResetResetResetResetResetResetDirection() override;
212
213     void resetResetResetResetResetResetResetResetAcceleration() override;
214
215     void resetResetResetResetResetResetResetResetDrag() override;
216
217     void resetResetResetResetResetResetResetResetIsMovingLeft() override;
218
219     void resetResetResetResetResetResetResetResetIsMovingRight() override;
220
221     void resetResetResetResetResetResetResetResetUpdate() override;
222
223     void resetResetResetResetResetResetResetResetHandleEvent() override;
224
225     void resetResetResetResetResetResetResetResetObserve() override;
226
227     void resetResetResetResetResetResetResetResetUnobserve() override;
228
229     void resetResetResetResetResetResetResetResetReset() override;
230
231     void resetResetResetResetResetResetResetResetResetVelocity() override;
232
233     void resetResetResetResetResetResetResetResetResetDirection() override;
234
235     void resetResetResetResetResetResetResetResetResetAcceleration() override;
236
237     void resetResetResetResetResetResetResetResetResetDrag() override;
238
239     void resetResetResetResetResetResetResetResetResetIsMovingLeft() override;
240
241     void resetResetResetResetResetResetResetResetResetIsMovingRight() override;
242
243     void resetResetResetResetResetResetResetResetResetUpdate() override;
244
245     void resetResetResetResetResetResetResetResetResetHandleEvent() override;
246
247     void resetResetResetResetResetResetResetResetResetObserve() override;
248
249     void resetResetResetResetResetResetResetResetResetUnobserve() override;
250
251     void resetResetResetResetResetResetResetResetResetReset() override;
252
253     void resetResetResetResetResetResetResetResetResetResetVelocity() override;
254
255     void resetResetResetResetResetResetResetResetResetResetDirection() override;
256
257     void resetResetResetResetResetResetResetResetResetResetAcceleration() override;
258
259     void resetResetResetResetResetResetResetResetResetResetDrag() override;
260
261     void resetResetResetResetResetResetResetResetResetResetIsMovingLeft() override;
262
263     void resetResetResetResetResetResetResetResetResetResetIsMovingRight() override;
264
265     void resetResetResetResetResetResetResetResetResetResetUpdate() override;
266
267     void resetResetResetResetResetResetResetResetResetResetHandleEvent() override;
268
269     void resetResetResetResetResetResetResetResetResetResetObserve() override;
270
271     void resetResetResetResetResetResetResetResetResetResetUnobserve() override;
272
273     void resetResetResetResetResetResetResetResetResetResetReset() override;
274
275     void resetResetResetResetResetResetResetResetResetResetResetVelocity() override;
276
277     void resetResetResetResetResetResetResetResetResetResetResetDirection() override;
278
279     void resetResetResetResetResetResetResetResetResetResetResetAcceleration() override;
280
281     void resetResetResetResetResetResetResetResetResetResetResetDrag() override;
282
283     void resetResetResetResetResetResetResetResetResetResetResetIsMovingLeft() override;
284
285     void resetResetResetResetResetResetResetResetResetResetResetIsMovingRight() override;
286
287     void resetResetResetResetResetResetResetResetResetResetResetUpdate() override;
288
289     void resetResetResetResetResetResetResetResetResetResetResetHandleEvent() override;
290
291     void resetResetResetResetResetResetResetResetResetResetResetObserve() override;
292
293     void resetResetResetResetResetResetResetResetResetResetResetUnobserve() override;
294
295     void resetResetResetResetResetResetResetResetResetResetResetReset() override;
296
297     void resetResetResetResetResetResetResetResetResetResetResetResetVelocity() override;
298
299     void resetResetResetResetResetResetResetResetResetResetResetResetDirection() override;
300
301     void resetResetResetResetResetResetResetResetResetResetResetResetAcceleration() override;
302
303     void resetResetResetResetResetResetResetResetResetResetResetResetDrag() override;
304
305     void resetResetResetResetResetResetResetResetResetResetResetResetIsMovingLeft() override;
306
307     void resetResetResetResetResetResetResetResetResetResetResetResetIsMovingRight() override;
308
309     void resetResetResetResetResetResetResetResetResetResetResetResetUpdate() override;
310
311     void resetResetResetResetResetResetResetResetResetResetResetResetHandleEvent() override;
312
313     void resetResetResetResetResetResetResetResetResetResetResetResetObserve() override;
314
315     void resetResetResetResetResetResetResetResetResetResetResetResetUnobserve() override;
316
317     void resetResetResetResetResetResetResetResetResetResetResetResetReset() override;
318
319     void resetResetResetResetResetResetResetResetResetResetResetResetResetVelocity() override;
320
321     void resetResetResetResetResetResetResetResetResetResetResetResetResetDirection() override;
322
323     void resetResetResetResetResetResetResetResetResetResetResetResetResetAcceleration() override;
324
325     void resetResetResetResetResetResetResetResetResetResetResetResetResetDrag() override;
326
327     void resetResetResetResetResetResetResetResetResetResetResetResetResetIsMovingLeft() override;
328
329     void resetResetResetResetResetResetResetResetResetResetResetResetResetIsMovingRight() override;
330
331     void resetResetResetResetResetResetResetResetResetResetResetResetResetUpdate() override;
332
333     void resetResetResetResetResetResetResetResetResetResetResetResetResetHandleEvent() override;
334
335     void resetResetResetResetResetResetResetResetResetResetResetResetResetObserve() override;
336
337     void resetResetResetResetResetResetResetResetResetResetResetResetResetUnobserve() override;
338
339     void resetResetResetResetResetResetResetResetResetResetResetResetResetReset() override;
340
341     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetVelocity() override;
342
343     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetDirection() override;
344
345     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetAcceleration() override;
346
347     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetDrag() override;
348
349     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetIsMovingLeft() override;
350
351     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetIsMovingRight() override;
352
353     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetUpdate() override;
354
355     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetHandleEvent() override;
356
357     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetObserve() override;
358
359     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetUnobserve() override;
360
361     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetReset() override;
362
363     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetResetVelocity() override;
364
365     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetResetDirection() override;
366
367     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetResetAcceleration() override;
368
369     void resetResetResetResetResetResetResetResetResetResetResetResetResetResetResetDrag() override;
369

```

```

34
35     ~Player() override = default;
36
37     Model::Type getType() const override { return Model::Type::ePlayer; }
38
39     void move(bool collision) override;
40
41     const std::pair<float, float>& getVelocity() const;
42
43     void setVelocity(const std::pair<float, float>& velocity);
44
45     const std::pair<float, float>& getDirection() const;
46
47     void setDirection(const std::pair<float, float>& direction);
48
49     const float getMaxVelocity() const;
50
51     const float getMaxAcceleration() const;
52
53     const float getDrag() const;
54
55     void setDrag(float drag);
56
57     bool isMovingUp() const;
58
59     void setIsMovingUp(bool isMovingUp);
60
61     bool isMovingDown() const;
62
63     void setIsMovingDown(bool isMovingDown);
64
65     bool isMovingLeft() const;
66
67     void setIsMovingLeft(bool isMovingLeft);
68
69     bool isMovingRight() const;
70
71     void setIsMovingRight(bool isMovingRight);
72
73     // TODO - visitor pattern
74     void accept(const std::shared_ptr<Visitor::IVisitor>& visitor) override { visitor->visit(*this); }
75
76     void onTrigger(EventType type, const std::shared_ptr<Event>& event) override {
77         event->send(*this); }
78
79     void handleEvent(const StopBonusEvent& event) override
80     {
81         std::cout << "lol\n";
82         accept(event.getBonus());
83     };
84 } // namespace Model
85
86 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_PLAYER_H

```

8.16 Spring.h

```

1 //
2 // Created by Pablo Deputter on 06/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_SPRING_H
6 #define DOODLEJUMP_SPRING_H
7
8 #include "Player.h"
9 #include <cmath>
10
11 namespace Model {
12
13     class Spring : public Entity
14     {
15     private:
16         std::pair<float, float> mBounds;
17         bool mMovingDown;
18         bool mInit;
19
20     public:
21         Spring() : Entity(15, .75f){};
22
23         ~Spring() override = default;
24
25         [[nodiscard]] Model::Type getType() const override;

```

```

26
27     void move(bool collision) override;
28
29     void initBounds();
30
31     void visit(Model::Player& player) override
32     {
33         // Original jump with default speed
34         float jumpPeak = (player.getMaxVelocity() / player.getDrag()) * (player.getMaxVelocity()
/ 2.f);
35         // Calculate speed needed for a jump with a vertical distance x5
36         float newSpeed = sqrt((jumpPeak * 5) * (player.getDrag() * 2.f));
37         player.setVelocity({player.getVelocity().first, newSpeed});
38     }
39
40     bool isBonus() const override { return true; }
41
42     bool isRemovable() const override { return mRemoveFlag; }
43 };
44 } // namespace Model
45
46 #endif // DOODLEJUMP_SPRING_H

```

8.17 StaticPlatform.h

```

1 //
2 // Created by Pablo Deputter on 05/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_STATICPLATFORM_H
6 #define DOODLEJUMP_STATICPLATFORM_H
7
8 #include "Entity.h"
9
10 namespace Model {
11
12 class StaticPlatform : public Entity
13 {
14
15 public:
16     StaticPlatform() : Entity(2, .70f) {}
17
18     ~StaticPlatform() override = default;
19
20     Model::Type getType() const override;
21
22     void move(bool collision) override;
23 };
24 } // namespace Model
25
26 #endif // DOODLEJUMP_STATICPLATFORM_H

```

8.18 TemporaryPlatform.h

```

1 //
2 // Created by Pablo Deputter on 05/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_TEMPORARYPLATFORM_H
6 #define DOODLEJUMP_TEMPORARYPLATFORM_H
7
8 #include "Entity.h"
9
10 namespace Model {
11
12 class TemporaryPlatform : public Entity
13 {
14
15 public:
16     TemporaryPlatform() : Entity(10, 0.10f) {}
17
18     ~TemporaryPlatform() override = default;
19
20     Model::Type getType() const override;
21
22     void move(bool collision) override;
23 };
24 } // namespace Model
25
26 #endif // DOODLEJUMP_TEMPORARYPLATFORM_H

```

8.19 VerticalPlatform.h

```

1 //
2 // Created by Pablo Deputter on 05/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_VERTICALPLATFORM_H
6 #define DOODLEJUMP_VERTICALPLATFORM_H
7
8 #include "Entity.h"
9 #include "util/Random.h"
10
11 namespace Model {
12
13 class VerticalPlatform : public Entity
14 {
15 private:
16     std::pair<float, float> mBounds;
17     bool mMovingDown;
18     bool mInit;
19
20 public:
21     VerticalPlatform() : Entity(7, .10f), mBounds({0.f, 0.f}), mMovingDown(true), mInit(false) {}
22
23     ~VerticalPlatform() override = default;
24
25     Model::Type getType() const override;
26
27     void move(bool collision) override;
28
29     void initBounds();
30 };
31 } // namespace Model
32
33 #endif // DOODLEJUMP_VERTICALPLATFORM_H

```

8.20 Observer.h

```

1 //
2 // Created by Pablo Deputter on 14/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_OBSERVER_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_OBSERVER_H
7
8 #include "Event.h"
9
10 #include <memory>
11
12 namespace Observer {
13
14 // template<class EventType, class DataType>
15 class Observer
16 {
17 public:
18     Observer() = default;
19     virtual ~Observer() = default;
20     virtual void onTrigger(EventType type, const std::shared_ptr<Event>& event) = 0;
21 };
22 } // namespace Observer
23
24 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_OBSERVER_H

```

8.21 Score.h

```

1 //
2 // Created by Pablo Deputter on 07/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_SCORE_H
6 #define DOODLEJUMP_SCORE_H
7
8 #include "Event.h"
9 #include "model/Entity.h"
10 #include <Subject.h>
11
12 namespace Model {
13
14 class Score : public Entity, public Observer::Observer, IEventHandler
15 {

```

```

16 private:
17     std::weak_ptr<Model::Entity> mLastCollision;
18
19 public:
20     Score() = default;
21
22     ~Score() override = default;
23
24     [[nodiscard]] Model::Type getType() const override { return Model::eScore; }
25
26     void setScore(unsigned int score) { Score::mScore += score; }
27
28     void onTrigger(EventType type, const std::shared_ptr<Event>& event) override {
29         event->send(*this); }
30
31     void handleEvent(const NewMaxHeightEvent& event) override;
32
33     void handleEvent(const CollisionEvent& event) override;
34
35     void move(bool collision) override {}
36
37     unsigned int getScore() const override { return mScore; }
38 };
39 // namespace Model
40 #endif // DOODLEJUMP_SCORE_H

```

8.22 Settings.h

```

1 //
2 // Created by Pablo Deputter on 12/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_SETTINGS_H
6 #define DOODLEJUMP_SETTINGS_H
7
8 namespace Settings {
9     static float CHANCE_STATIC = .9;
10    static float CHANCE_HORIZONTAL = .04;
11    static float CHANCE_VERTICAL = .04;
12    static float CHANCE_TEMPORARY = .02f;
13    static float CHANCE_BONUS = .1f;
14    static float CHANCE_SPRING = 1.;
15    static float CHANCE_JETPACK = .0f;
16    static unsigned int MIN_PLATFORMS = 7;
17    static unsigned int MAX_PLATFORMS = 20;
18    static float DIFFICULTY = 0.f;
19    enum Difficulty
20    {
21        eEasy = 0,
22        eNormal,
23        eDifficult,
24        eHard,
25        eExtreme
26    };
27    bool static setDifficulty(Difficulty difficulty)
28    {
29        switch (difficulty) {
30            case eEasy:
31                CHANCE_STATIC = .9f;
32                CHANCE_HORIZONTAL = .04f;
33                CHANCE_VERTICAL = .04f;
34                CHANCE_TEMPORARY = .02f;
35
36                CHANCE_BONUS = .05f;
37                CHANCE_SPRING = .95f;
38                CHANCE_JETPACK = .05f;
39
40                MAX_PLATFORMS = 20;
41                DIFFICULTY = 0.f;
42                break;
43            case eNormal:
44                CHANCE_STATIC = .8f;
45                CHANCE_HORIZONTAL = .07f;
46                CHANCE_VERTICAL = .07f;
47                CHANCE_TEMPORARY = .06f;
48
49                CHANCE_BONUS = .15f;
50                CHANCE_SPRING = .9f;
51                CHANCE_JETPACK = .1f;
52
53                MAX_PLATFORMS = 17;
54                DIFFICULTY = 0.25f;
55        }
56    }
57 }
58 #endif

```

```

78         break;
79     case eDifficult:
80         CHANCE_STATIC = .6f;
81         CHANCE_HORIZONTAL = .15f;
82         CHANCE_VERTICAL = .15f;
83         CHANCE_TEMPORARY = .1f;
84
85         CHANCE_BONUS = .25f;
86         CHANCE_SPRING = .75f;
87         CHANCE_JETPACK = .25f;
88
89         MAX_PLATFORMS = 14;
90         DIFFICULTY = 0.5f;
91         break;
92     case eHard:
93         CHANCE_STATIC = .4f;
94         CHANCE_HORIZONTAL = .22f;
95         CHANCE_VERTICAL = .22f;
96         CHANCE_TEMPORARY = .16f;
97
98         CHANCE_BONUS = .30f;
99         CHANCE_SPRING = .75f;
100        CHANCE_JETPACK = .25f;
101
102        MAX_PLATFORMS = 10;
103        DIFFICULTY = 0.75f;
104        break;
105    case eExtreme:
106        CHANCE_STATIC = .2f;
107        CHANCE_HORIZONTAL = .3f;
108        CHANCE_VERTICAL = .3f;
109        CHANCE_TEMPORARY = .2f;
110
111        CHANCE_BONUS = .30f;
112        CHANCE_SPRING = .75f;
113        CHANCE_JETPACK = .25f;
114
115        MAX_PLATFORMS = 7;
116        DIFFICULTY = 0.85f;
117        break;
118    }
119    return true;
120 }
121 } // namespace Settings
122
123 #endif // DOODLEJUMP_SETTINGS_H

```

8.23 Subject.h

```

1 //
2 // Created by Pablo Deputter on 14/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_SUBJECT_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_SUBJECT_H
7
8 #include "Observer.h"
9
10 #include <algorithm>
11 #include <memory>
12 #include <vector>
13
14 namespace Observer {
15     // template<class EventType, class Event>
16     class Subject
17     {
18     private:
19         std::vector<std::shared_ptr<Observer>> mObservers;
20     public:
21         Subject() = default;
22         virtual ~Subject() = default;
23         void add(const std::shared_ptr<Observer>& observer) { mObservers.emplace_back(observer); }
24
25         [[nodiscard]] const std::vector<std::shared_ptr<Observer>>& getObservers() const { return
26             mObservers; }
27         void clear() { mObservers.clear(); }
28         void trigger(EventType type, const std::shared_ptr<Event>& event) const
29         {
30             for (const auto& i : mObservers) {
31                 i->onTrigger(type, event);
32             }
33         }
34     };
35 };

```

```

56 } // namespace Observer
57
58 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_SUBJECT_H

```

8.24 Camera.h

```

1 //
2 // Created by Pablo Deputter on 19/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_CAMERA_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_CAMERA_H
7
8 #include "Subject.h"
9
10 #include <memory>
11 #include <utility>
12
13 namespace Utils {
14     class Camera : public Observer::Subject
15     {
16     private:
17         float mWorldLeft{};
18         float mWorldRight{};
19         float mWorldTop{};
20         float mWorldBottom{};
21         float mWindowLeft{};
22         float mWindowRight{};
23         float mWindowTop{};
24         float mWindowBottom{};
25         float mCameraX{};
26         float mCameraY{};
27         float mLastMaxHeight;
28         float mMaxHeight;
29         Camera() : mLastMaxHeight(0.f), mMaxHeight(0.f) {}
30
31     public:
32         virtual ~Camera() = default;
33         Camera(const Camera&) = delete;
34         Camera& operator=(const Camera&) = delete;
35         static Camera& getInstance();
36
37         void reset();
38
39         [[nodiscard]] std::pair<float, float> getWorldDimensions() const;
40         void setWorldDimensions(float right, float top, float left = 0.f, float bottom = 0.f);
41         [[nodiscard]] std::pair<float, float> getWindowDimensions() const;
42         void setWindowDimensions(float right, float bottom, float left = 0.f, float top = 0.f);
43         [[nodiscard]] std::pair<float, float> transform(float x, float y, float left = 0.f, float top =
44             0.f) const;
45         [[nodiscard]] std::pair<float, float> inverseTransform(float x, float y) const;
46         void move(float x, float y);
47         [[nodiscard]] float getX() const { return mCameraX; }
48         [[nodiscard]] float getY() const { return mCameraY; }
49         [[nodiscard]] float getMaxHeight() const { return mMaxHeight; }
50         [[nodiscard]] float getLastMaxHeight() const { return mLastMaxHeight; }
51         bool isMaxHeight(float height);
52     };
53 } // namespace Utils
54
55 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_CAMERA_H

```

8.25 Exception.h

```

1 //
2 // Created by Pablo Deputter on 09/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_EXCEPTION_H
6 #define DOODLEJUMP_EXCEPTION_H
7
8 #include <exception>
9 #include <string>
10
11 namespace Utils {
12     class Exception : public std::exception
13     {
14     protected:

```

```

16         std::string mValue;
17
18     public:
19         explicit Exception(std::string value) : mValue(std::move(value)) {}
20
21         Exception() = default;
22
23         ~Exception() override = default;
24
25         [[nodiscard]] const char* what() const noexcept override { return mValue.c_str(); }
26 };
27
28 class FileException : public Exception
29 {
30     public:
31         explicit FileException(std::string file, std::string sort)
32             : Exception("Failed to load " + std::move(sort) + " '" + std::move(file) + "'. File is
33               missing.\n")
34         {
35         };
36 } // namespace Utils
37
38 #endif // DOODLEJUMP_EXCEPTION_H

```

8.26 Random.h

```

1 //
2 // Created by Pablo Deputter on 11/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_RANDOM_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_RANDOM_H
7
8 #include <random>
9
10 namespace Utils {
11     class Random
12     {
13     private:
14         Random() = default;
15
16     public:
17         ~Random() = default;
18         Random(const Random&) = delete;
19         Random& operator=(const Random&) = delete;
20         static Random& getInstance();
21         [[nodiscard]] float random(float a, float b);
22     };
23 } // namespace Utils
24
25 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_RANDOM_H

```

8.27 Stopwatch.h

```

1 //
2 // Created by Pablo Deputter on 14/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_STOPWATCH_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_STOPWATCH_H
7
8 #include "model/Entity.h"
9
10 #include <chrono>
11 #include <iostream>
12 #include <map>
13
14 enum Type
15 {
16     ePlayer = 0,
17     eBonus = 1,
18     eStatic = 2,
19     eHorizontal = 3,
20     eVertical = 4,
21     eTemporary = 5,
22     eBackground = 6,
23     eJetpack = 7,

```



```

25         eSpring = 8
26 };
27
31 namespace Utils {
32 class Stopwatch
33 {
34 private:
35     std::chrono::high_resolution_clock::time_point mTime;
36     float mDeltaTime{};
37     std::map<Type, std::pair<float, std::chrono::high_resolution_clock::time_point> mTimers;
38
39 public:
40     std::shared_ptr<Model::Entity> mPlayer;
41
42     Stopwatch() = default;
43
44 public:
45     ~Stopwatch() = default;
46     Stopwatch(const Stopwatch&) = delete;
47     Stopwatch& operator=(const Stopwatch&) = delete;
48     static Stopwatch& getInstance();
49     void start();
50     [[nodiscard]] float lap();
51     [[nodiscard]] float getDelta() const;
52
53     // TODO - jetpack
54     void addTimer(unsigned int key, float amount)
55     {
56         std::cout << "addTimer\n";
57         if (mTimers.find(Type(key)) != std::end(mTimers)) {
58             // std::cout << "ERROR\n";
59         }
60         mTimers[Type(key)] = {amount, std::chrono::high_resolution_clock::now()};
61     }
62     // std::chrono::duration<float> ms_delta = std::chrono::high_resolution_clock::now() -
63     mTime;
64     // // Reset / lap stopwatch
65     // mTime = std::chrono::high_resolution_clock::now();
66     // mDeltaTime = ms_delta.count();
67     // // Return milliseconds as float
68     // return mDeltaTime;
69     bool checkTimer(unsigned int key)
70     {
71         if (mTimers.find(Type(key)) == std::end(mTimers)) {
72             // std::cout << "ERROR\n";
73             return false;
74         }
75         auto val = mTimers[Type(key)];
76         std::chrono::duration<float> ms_delta = std::chrono::high_resolution_clock::now() -
77         val.second;
78         std::cout << ms_delta.count() << "\n";
79         if (ms_delta.count() > val.first) {
80             mTimers.erase(Type(key));
81             return false;
82         }
83         return true;
84     }
85 };
86 } // namespace Utils
87
88 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_STOPWATCH_H

```

8.28 Utilities.h

```

1 //
2 // Created by Pablo Deputter on 21/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_COLLISION_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_COLLISION_H
7
8 #include "model/Entity.h"
9
10 #include <cmath>
11
12 namespace Utilities {
13 class Utilities
14 {
15 public:
16     static bool checkCollision(const std::shared_ptr<Model::Entity>& l, const
17     std::shared_ptr<Model::Entity>& r);
18     static bool checkWeight(float& rand, float weight);
19 };
20

```

```

37 } // namespace Utils
38
39 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_COLLISION_H

```

8.29 World.h

```

1 //
2 // Created by Pablo Deputter on 18/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_WORLD_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_WORLD_H
7
8 #include "controller/IController.h"
9 #include "controller/PlayerController.h"
10
11 #include "model/Entity.h"
12 #include "model/Player.h"
13
14 #include "AbstractFactory.h"
15
16 #include "model/Jetpack.h"
17
18 #include "HighScore.h"
19
20 #include "Score.h"
21 #include "util/Camera.h"
22 #include "util/Random.h"
23 #include "util/Utilities.h"
24
25 #include "Settings.h"
26
27 #include "Event.h"
28
29 #include <iostream>
30 #include <memory>
31 #include <vector>
32
33 class World
34 {
35 private:
36     std::shared_ptr<Model::Player> mPlayer;
37     std::vector<std::shared_ptr<Model::Entity>> mEntities;
38     std::shared_ptr<Model::AbstractFactory> mFactory;
39     std::vector<std::shared_ptr<Model::Entity>> mBackground;
40     std::shared_ptr<Model::Score> mScore;
41     unsigned int mActivePlatforms;
42     Settings::Difficulty mDifficulty;
43
44     bool mPlaying;
45
46 public:
47     explicit World(std::shared_ptr<Model::AbstractFactory>& factory);
48
49     ~World() { destroy(); }
50     void initWorld();
51     void events(const std::string& move, bool isPressed) const;
52     void update();
53     void render() const;
54     void generateEntity();
55     void spawnPlatform(float x, float y);
56     void spawnBonus(float x, float y);
57     void spawnEntity(float x, float y, Model::Type type);
58     bool checkDifficulty();
59     void addEntity(const std::shared_ptr<Model::Entity>& entity);
60     void removeEntities();
61
62     void destroy();
63
64     [[nodiscard]] const std::shared_ptr<Model::Score>& getScore() const { return mScore; }
65
66     [[nodiscard]] bool isPlaying() const { return mPlaying; }
67 };
68
69 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_WORLD_H

```

8.30 ConcreteFactory.h

```

1 //

```

```

2 // Created by Pablo Deputter on 21/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_CONCRETEFACTORY_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_CONCRETEFACTORY_H
7
8 #include "AbstractFactory.h"
9
10 #include "model/Background.h"
11 #include "model/HorizontalPlatform.h"
12 #include "model/Jetpack.h"
13 #include "model/Platform.h"
14 #include "model/Player.h"
15 #include "model/Spring.h"
16 #include "model/StaticPlatform.h"
17 #include "model/TemporaryPlatform.h"
18 #include "model/VerticalPlatform.h"
19
20 #include "controller/BonusController.h"
21 #include "controller/PlatformController.h"
22 #include "controller/PlayerController.h"
23
24 #include "view/BackgroundView.h"
25 #include "view/BonusView.h"
26 #include "view/IView.h"
27 #include "view/PlatformView.h"
28 #include "view/PlayerView.h"
29 #include "view/ScoreView.h"
30
31 #include "Score.h"
32
33 #include "SFML/Graphics.hpp"
34
35 // TODO - attach View Observers to Entity Subjects
36
37 namespace View {
38 class ConcreteFactory : public Model::AbstractFactory
39 {
40 public:
41     ConcreteFactory() = default;
42
43     ConcreteFactory(const std::shared_ptr<sf::RenderWindow>& window) : mWindow(window) {}
44
45     ~ConcreteFactory() override = default;
46
47     std::shared_ptr<Model::Player> createPlayer() override;
48
49     std::shared_ptr<Model::Entity> createStaticPlatform() override;
50
51     std::shared_ptr<Model::Entity> createHorizontalPlatform() override;
52
53     std::shared_ptr<Model::Entity> createVerticalPlatform() override;
54
55     std::shared_ptr<Model::Entity> createTemporaryPlatform() override;
56
57     std::shared_ptr<Model::Entity> createSpring() override;
58
59     std::shared_ptr<Model::Entity> createJetpack() override;
60
61     std::shared_ptr<Model::Entity> createBackground() override;
62
63     std::shared_ptr<Model::Score> createScore() override;
64
65 private:
66     std::shared_ptr<sf::RenderWindow> mWindow;
67 };
68 } // namespace View
69
70 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_CONCRETEFACTORY_H

```

8.31 Game.h

```

1 //
2 // Created by Pablo Deputter on 14/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_GAME_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_GAME_H
7
8 #include "ConcreteFactory.h"
9
10 #include "World.h"
11

```

```

12 #include "util/Resourcemanager.h"
13 #include "util/Stopwatch.h"
14
15 #include "SFML/Graphics.hpp"
16 #include "SFML/Window.hpp"
17
21 class Game
22 {
23     std::shared_ptr<sf::RenderWindow> mWindow;
24     std::unique_ptr<World> mWorld;
25     std::shared_ptr<Model::AbstractFactory> mFactory;
26 public:
27     Game() = default;
28
29     explicit Game(unsigned int width = 800, unsigned int height = 1440);
30
31     ~Game() = default;
32
33     void initializeResources();
34
35     void processEvents();
36
37     void handlePlayerInput(sf::Keyboard::Key key, bool isPressed);
38
39     void render();
40
41     void run();
42
43     void drawHighScoreTable();
44 };
45
46 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_GAME_H

```

8.32 Resourcemanager.h

```

1 //
2 // Created by Pablo Deputter on 29/11/2021.
3 //
4
5 #ifndef DOODLEJUMP_RESOURCEMANAGER_H
6 #define DOODLEJUMP_RESOURCEMANAGER_H
7
8 #include "SFML/Graphics.hpp"
9 #include "SFML/Audio.hpp"
10
11
12 #include "model/Entity.h"
13 #include "util/Exception.h"
14
15 #include <map>
16 #include <string>
17 #include <utility>
18 #include <typeinfo>
19
20 namespace Utils {
21 template<class Type>
22 struct Resourceholder
23 {
24 private:
25     std::string mPath;
26     std::map<Model::Type, std::shared_ptr<Type> > mResources;
27 public:
28     explicit Resourceholder(std::string path) : mPath(std::move(path)) {}
29
30     ~Resourceholder() = default;
31
32     void insert(Model::Type type, const std::string& subPath)
33     {
34         std::shared_ptr<Type> file = std::make_shared<Type>();
35         std::string path = mPath + std::string(subPath);
36
37         if (!file->loadFromFile(path)) {
38             throw(FileException(path, typeid(file).name()));
39         }
40         mResources.emplace(type, file);
41     }
42
43     std::shared_ptr<Type>& get(Model::Type type) { return mResources[type]; }
44 };
45
46 class Resourcemanager
47 {
48 private:

```

```

49     std::shared_ptr<Resourceholder<sf::Texture>> mTextures;
50     std::shared_ptr<Resourceholder<sf::Font>> mFonts;
51     std::shared_ptr<Resourceholder<sf::SoundBuffer>> mSounds;
52
53     explicit Resourcemanager(const std::string& path)
54     {
55         mTextures = std::make_shared<Resourceholder<sf::Texture>>(path);
56         mFonts = std::make_shared<Resourceholder<sf::Font>>(path);
57         mSounds = std::make_shared<Resourceholder<sf::SoundBuffer>>(path);
58     }
59
60 public:
61     ~Resourcemanager() = default;
62
63     Resourcemanager(const Resourcemanager&) = delete;
64
65     Resourcemanager& operator=(const Resourcemanager&) = delete;
66
67     static Resourcemanager& getInstance()
68     {
69         static Resourcemanager instance(
70             "/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/resource");
71         return instance;
72     }
73
74     void addTexture(Model::Type type, const std::string& subPath) { mTextures->insert(type, subPath); }
75
76     void addFont(Model::Type type, const std::string& subPath) { mFonts->insert(type, subPath); }
77
78     void addSound(Model::Type type, const std::string& subPath) { mSounds->insert(type, subPath); }
79
80     [[nodiscard]] const std::shared_ptr<Resourceholder<sf::Texture>>& getTextures() const { return
mTextures; }
81
82     [[nodiscard]] const std::shared_ptr<Resourceholder<sf::Font>>& getFonts() const { return mFonts; }
83
84     [[nodiscard]] const std::shared_ptr<Resourceholder<sf::SoundBuffer>>& getSounds() const { return
mSounds; }
85 };
86 } // namespace Utils
87
88 #endif // DOODLEJUMP_RESOURCEMANAGER_H

```

8.33 BackgroundView.h

```

1 //
2 // Created by Pablo Deputter on 29/11/2021.
3 //
4
5 #ifndef DOODLEJUMP_BACKGROUNDVIEW_H
6 #define DOODLEJUMP_BACKGROUNDVIEW_H
7
8 #include "IView.h"
9
10 #include <filesystem>
11
12 namespace View {
13
14 class BackgroundView : public IView
15 {
16 public:
17     BackgroundView(const std::shared_ptr<Model::Entity>& entity, const
std::shared_ptr<sf::RenderWindow>& window)
18         : IView(entity, window)
19     {
20         std::shared_ptr<sf::Texture>& tex =
Utils::Resourcemanager::getInstance().getTextures()->get(Model::eBackground);
21
22         mSprite = std::make_unique<sf::Sprite>();
23         mSprite->setTexture(*tex);
24         mSprite->scale(1.f, 1.f);
25         mSprite->setColor(sf::Color(255, 255, 255, 255));
26
27         mEntity->setWidth((float)tex->getSize().x * mSprite->getScale().x);
28         mEntity->setHeight((float)tex->getSize().y * mSprite->getScale().y);
29     }
30
31     ~BackgroundView() override = default;
32
33     void handleEvent(const DrawEvent& event) override;
34
35     void handleEvent(const OutOfViewEvent& event) override;
36

```

```

37 };
38 } // namespace View
39
40 #endif // DOODLEJUMP_BACKGROUNDVIEW_H

```

8.34 BonusView.h

```

1 //
2 // Created by Pablo Deputter on 06/12/2021.
3 //
4
5 #ifndef DOODLEJUMP_BONUSVIEW_H
6 #define DOODLEJUMP_BONUSVIEW_H
7
8 #include "IView.h"
9
10 namespace View {
11
12 class BonusView : public IView
13 {
14 public:
15     BonusView(const std::shared_ptr<Model::Entity>& entity, const std::shared_ptr<sf::RenderWindow>&
        window)
16         : IView(entity, window)
17     {
18         Model::Type type = entity->getType();
19         std::shared_ptr<sf::Texture>& tex =
            Utils::ResourceManager::getInstance().getTextures()->get(type);
20
21         mSprite = std::make_unique<sf::Sprite>();
22         mSprite->setTexture(*tex);
23         mSprite->scale(.1f, .1f);
24
25         auto texSize = Utils::Camera::getInstance().inverseTransform(
26             (float)tex->getSize().x * mSprite->getScale().x, (float)tex->getSize().y *
            mSprite->getScale().y);
27
28         mEntity->setWidth(texSize.first);
29         mEntity->setHeight(Utils::Camera::getInstance().getWorldDimensions().second -
            texSize.second);
30     }
31
32     BonusView() = default;
33
34     ~BonusView() override = default;
35 };
36 } // namespace View
37
38 #endif // DOODLEJUMP_BONUSVIEW_H

```

8.35 IView.h

```

1 //
2 // Created by Pablo Deputter on 19/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_IVIEW_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_IVIEW_H
7
8 #include "model/Bonus.h"
9 #include "model/Player.h"
10
11 #include "Observer.h"
12 #include "util/Camera.h"
13
14 #include "util/ResourceManager.h"
15
16 #include "SFML/Graphics.hpp"
17
18 #include "Event.h"
19
20 #include <iostream>
21 #include <memory>
22
23 namespace View {
24
25 class IView : public Observer::Observer, public IEventHandler, public
        std::enable_shared_from_this<Observer::Observer>
26 {

```

```

27 protected:
28     std::shared_ptr<Model::Entity> mEntity;
29     std::unique_ptr<sf::Sprite> mSprite;
30     std::shared_ptr<sf::RenderWindow> mWindow;
31     std::unique_ptr<sf::Sound> mSound;
32
33 public:
34     IView(const std::shared_ptr<Model::Entity>& entity, const std::shared_ptr<sf::RenderWindow>&
        window);
35
36     IView() = default;
37
38     virtual ~IView() = default;
39
40     void drawCollisionBox();
41
42     void onTrigger(EventType type, const std::shared_ptr<Event>& event) override;
43
44     void handleEvent(const DrawEvent& event) override;
45
46     void handleEvent(const OutOfViewEvent& event) override;
47
48     virtual void handleEvent(const CollisionEvent& event) override {}
49
50     void handleEvent(const NewDifficultyEvent& event) override {}
51
52     template <class Type>
53     static void setRainbowColor(const std::unique_ptr<sf::Text>& object)
54     {
55         if (object->getFillColor().r + 5 <= 255 && object->getFillColor().g == 0 &&
56             object->getFillColor().b == 0) {
57             object->setFillColor(sf::Color(object->getFillColor().r + 5, 0, 0));
58         } else if (object->getFillColor().r == 255 && object->getFillColor().g + 5 <= 255 &&
59             object->getFillColor().b == 0) {
60             object->setFillColor(sf::Color(255, object->getFillColor().g + 5, 0));
61         } else if (object->getFillColor().r - 5 >= 0 && object->getFillColor().g == 255 &&
62             object->getFillColor().b == 0) {
63             object->setFillColor(sf::Color(object->getFillColor().r - 5, 255, 0));
64         } else if (object->getFillColor().r == 0 && object->getFillColor().g == 255 &&
65             object->getFillColor().b + 5 <= 255) {
66             object->setFillColor(sf::Color(0, 255, object->getFillColor().b + 5));
67         } else if (object->getFillColor().r == 0 && object->getFillColor().g - 5 >= 0 &&
68             object->getFillColor().b == 255) {
69             object->setFillColor(sf::Color(0, object->getFillColor().g - 5, 255));
70         } else if (object->getFillColor().r + 5 <= 255 && object->getFillColor().g == 0 &&
71             object->getFillColor().b == 255) {
72             object->setFillColor(sf::Color(object->getFillColor().r + 5, 0, 255));
73         } else if (object->getFillColor().r == 255 && object->getFillColor().g == 0 &&
74             object->getFillColor().b - 5 >= 0) {
75             object->setFillColor(sf::Color(255, 0, object->getFillColor().b - 5));
76         }
77     }
78 };
79 } // namespace View
80
81 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_IVIEW_H

```

8.36 PlatformView.h

```

1 //
2 // Created by Pablo Deputter on 19/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_PLATFORMVIEW_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_PLATFORMVIEW_H
7
8 #include "IView.h"
9 #include <thread>
10
11 namespace View {
12
13 class PlatformView : public IView
14 {
15 public:
16     PlatformView(const std::shared_ptr<Model::Entity>& entity, const
        std::shared_ptr<sf::RenderWindow>& window)
17         : IView(entity, window)
18     {
19         Model::Type type = entity->getType();
20         std::shared_ptr<sf::Texture>& tex =
            Utils::ResourceManager::getInstance().getTextures()->get(type);
21
22         mSprite->setTexture(*tex);

```

```

23         mSprite->scale(.25f, .25f);
24
25         // TODO - sound
26         //
27         mSound->setBuffer(*Utils::ResourceManager::getInstance().getSounds()->get(mEntity->getType()));
28         //
29         // mSound->setVolume(100.f);
30
31         auto texSize = Utils::Camera::getInstance().inverseTransform(
32             (float)tex->getSize().x * mSprite->getScale().x, (float)tex->getSize().y *
33             mSprite->getScale().y);
34
35         mEntity->setWidth(texSize.first);
36         mEntity->setHeight(Utils::Camera::getInstance().getWorldDimensions().second -
37             texSize.second);
38     }
39
40     PlatformView() = default;
41
42     ~PlatformView() override = default;
43
44     void handleEvent(const CollisionEvent& event) override
45     {
46         // TODO - threads leaks
47         // mSound->play();
48     }
49 };
50 } // namespace View
51
52 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_PLATFORMVIEW_H

```

8.37 PlayerView.h

```

1 //
2 // Created by Pablo Deputter on 19/11/2021.
3 //
4
5 #ifndef ADVANCED_PROGRAMMING_DOODLEJUMP_PLAYERVIEW_H
6 #define ADVANCED_PROGRAMMING_DOODLEJUMP_PLAYERVIEW_H
7
8 #include "IView.h"
9
10 namespace View {
11
12 class PlayerView : public IView
13 {
14 public:
15     PlayerView(const std::shared_ptr<Model::Entity>& entity, const std::shared_ptr<sf::RenderWindow>&
16         window)
17         : IView(entity, window)
18     {
19         std::shared_ptr<sf::Texture>& tex =
20             Utils::ResourceManager::getInstance().getTextures()->get(Model::ePlayer);
21
22         mSprite = std::make_unique<sf::Sprite>();
23         mSprite->setTexture(*tex);
24         mSprite->scale(2.f, 2.f);
25
26         auto texSize = Utils::Camera::getInstance().inverseTransform(
27             (float)tex->getSize().x * mSprite->getScale().x, (float)tex->getSize().y *
28             mSprite->getScale().y);
29
30         mEntity->setWidth(texSize.first);
31         mEntity->setHeight(Utils::Camera::getInstance().getWorldDimensions().second -
32             texSize.second);
33     }
34
35     PlayerView() = default;
36
37     ~PlayerView() override = default;
38 };
39 } // namespace View
40
41 #endif // ADVANCED_PROGRAMMING_DOODLEJUMP_PLAYERVIEW_H

```

8.38 ScoreView.h

```

1 //
2 // Created by Pablo Deputter on 09/12/2021.

```



```
3 //
4
5 #ifndef DOODLEJUMP_SCOREVIEW_H
6 #define DOODLEJUMP_SCOREVIEW_H
7
8 #include "IView.h"
9
10 #include "Score.h"
11
12 namespace View {
13
14 class ScoreView : public IView
15 {
16 private:
17     std::unique_ptr<sf::Text> mText;
18     std::unique_ptr<sf::Text> mDiffText;
19
20     void handleEvent(const DrawEvent& event) override;
21
22     void handleEvent(const NewDifficultyEvent& event) override;
23
24 public:
25     ScoreView(std::shared_ptr<Model::Score>& entity, const std::shared_ptr<sf::RenderWindow>& window)
26         : IView(entity, window)
27     {
28         std::shared_ptr<sf::Font>& font =
29             Utils::ResourceManager::getInstance().getFonts()->get(Model::eScore);
30
31         mSprite = std::make_unique<sf::Sprite>();
32         mText = std::make_unique<sf::Text>();
33         mText->setFont(*font);
34         mText->setFillColor(sf::Color::Black);
35
36         mDiffText = std::make_unique<sf::Text>();
37         mDiffText->setFont(*font);
38         mDiffText->setFillColor(sf::Color::Black);
39     }
40
41     ~ScoreView() override = default;
42 };
43 // namespace View
44 #endif // DOODLEJUMP_SCOREVIEW_H
```


Index

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/AbstractFactory.h, 77

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Event.h, 79

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/HighScore.h, 82

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/IVisitor.h, 83

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Observer.h, 90

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Score.h, 90

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Settings.h, 91

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/Subject.h, 92

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/World.h, 96

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/BonusController.h, 77

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/IController.h, 78

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/PlatformController.h, 78

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/controller/PlayerController.h, 79

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Background.h, 83

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Bonus.h, 83

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Entity.h, 84

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/HorizontalPlatform.h, 85

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Jetpack.h, 86

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Platform.h, 87

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Player.h, 87

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/Spring.h, 88

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/StaticPlatform.h, 89

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/TemporaryPlatform.h, 89

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/model/VerticalPlatform.h, 90

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/Camera.h, 93

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/Exception.h, 93

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/Random.h, 94

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/Stopwatch.h, 94

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/engine/include/util/Utilities.h, 95

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/ConcreteFactory.h, 96

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/Game.h, 97

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/util/ResourceManager.h, 98

/Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/BackgroundView.h, 99

- 99
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/BonusView.h, 100
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/IView.h, 100
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/PlatformView.h, 101
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/PlayerView.h, 102
- /Users/pablodeputter/Documents/GitHub/Advanced-Programming-DoodleJump/game/include/view/ScoreView.h, 102
- accept
 - Model::Player, 54
- add
 - Observer::Subject, 68
- addEntity
 - World, 73
- checkCollision
 - Utils::Utilities, 70
- checkWeight
 - Utils::Utilities, 71
- CollisionEvent, 25
 - send, 26
- Controller::BonusController, 19
- Controller::IController, 39
 - handleEvent, 40
 - onTrigger, 40
- Controller::PlatformController, 51
- Controller::PlayerController, 55
 - handleEvent, 55, 56
- createBackground
 - View::ConcreteFactory, 27
- createHorizontalPlatform
 - View::ConcreteFactory, 27
- createJetpack
 - View::ConcreteFactory, 27
- createPlayer
 - View::ConcreteFactory, 27
- createScore
 - View::ConcreteFactory, 27
- createSpring
 - View::ConcreteFactory, 27
- createStaticPlatform
 - View::ConcreteFactory, 28
- createTemporaryPlatform
 - View::ConcreteFactory, 28
- createVerticalPlatform
 - View::ConcreteFactory, 28
- Difficulty
 - Settings, 13
- DrawEvent, 28
- send, 29
- EventArgs, 35
- events
 - World, 73
- Game, 36
- getDelta
 - Utils::Stopwatch, 66
- getHeight
 - Model::Entity, 31
- getInstance
 - Utils::Camera, 21
 - Utils::Random, 57
 - Utils::Stopwatch, 66
- getMaxHeight
 - Utils::Camera, 21
- getMSort
 - Model::Bonus, 18
- getScore
 - Model::Score, 60
- getType
 - Model::Background, 16
 - Model::Bonus, 18
 - Model::Entity, 31
 - Model::HorizontalPlatform, 38
 - Model::Jetpack, 44
 - Model::Platform, 50
 - Model::Player, 54
 - Model::Score, 60
 - Model::Spring, 62
 - Model::StaticPlatform, 64
 - Model::TemporaryPlatform, 69
 - Model::VerticalPlatform, 72
- getWindowDimensions
 - Utils::Camera, 21
- getWorldDimensions
 - Utils::Camera, 22
- getX
 - Model::Entity, 31
 - Utils::Camera, 22
- getY
 - Model::Entity, 31
 - Utils::Camera, 22
- handleEvent
 - Controller::IController, 40
 - Controller::PlayerController, 55, 56
 - Model::Player, 54
 - Model::Score, 60
 - View::BackgroundView, 17
 - View::IView, 42
 - View::PlatformView, 52
- HighScore, 37
- HighScoreScore, 37
- IEventHandler, 41
- inverseTransform
 - Utils::Camera, 22

- isBonus
 - Model::Jetpack, 44
 - Model::Spring, 62
- isMaxHeight
 - Utils::Camera, 23
- isRemovable
 - Model::Jetpack, 45
 - Model::Spring, 62
- KeyPressedEvent, 45
 - send, 46
- lap
 - Utils::Stopwatch, 66
- mHeight
 - Model::Entity, 34
- Model, 11
 - Type, 12
- Model::AbstractFactory, 15
- Model::Background, 15
 - getType, 16
 - move, 16
- Model::Bonus, 17
 - getMSort, 18
 - getType, 18
 - setMSort, 18
- Model::CollisionBox, 25
- Model::Entity, 29
 - getHeight, 31
 - getType, 31
 - getX, 31
 - getY, 31
 - mHeight, 34
 - move, 32
 - mWidth, 34
 - mX, 34
 - mY, 34
 - setHeight, 32
 - setWidth, 33
 - setX, 33
 - setY, 33
 - visit, 33
- Model::HorizontalPlatform, 38
 - getType, 38
 - move, 39
- Model::Jetpack, 44
 - getType, 44
 - isBonus, 44
 - isRemovable, 45
 - move, 45
 - visit, 45
- Model::Platform, 50
 - getType, 50
 - move, 51
- Model::Player, 53
 - accept, 54
 - getType, 54
 - handleEvent, 54
 - move, 54
 - onTrigger, 54
- Model::Score, 59
 - getScore, 60
 - getType, 60
 - handleEvent, 60
 - move, 60
 - onTrigger, 61
- Model::Spring, 62
 - getType, 62
 - isBonus, 62
 - isRemovable, 62
 - move, 63
 - visit, 63
- Model::StaticPlatform, 63
 - getType, 64
 - move, 64
- Model::TemporaryPlatform, 69
 - getType, 69
 - move, 69
- Model::VerticalPlatform, 71
 - getType, 72
 - move, 72
- move
 - Model::Background, 16
 - Model::Entity, 32
 - Model::HorizontalPlatform, 39
 - Model::Jetpack, 45
 - Model::Platform, 51
 - Model::Player, 54
 - Model::Score, 60
 - Model::Spring, 63
 - Model::StaticPlatform, 64
 - Model::TemporaryPlatform, 69
 - Model::VerticalPlatform, 72
 - Utils::Camera, 23
- MoveEvent, 46
 - send, 46
- mWidth
 - Model::Entity, 34
- mX
 - Model::Entity, 34
- mY
 - Model::Entity, 34
- NewDifficultyEvent, 47
 - send, 47
- NewMaxHeightEvent, 48
 - send, 48
- Observer, 12
- Observer::Observer, 48
 - onTrigger, 49
- Observer::Subject, 67
 - add, 68
- onTrigger
 - Controller::IController, 40
 - Model::Player, 54
 - Model::Score, 61

- Observer::Observer, 49
- View::IView, 42
- operator=
 - Utils::Camera, 23
 - Utils::Random, 57
 - Utils::Stopwatch, 67
- OutOfViewEvent, 49
 - send, 50
- random
 - Utils::Random, 58
- send
 - CollisionEvent, 26
 - DrawEvent, 29
 - KeyPressedEvent, 46
 - MoveEvent, 46
 - NewDifficultyEvent, 47
 - NewMaxHeightEvent, 48
 - OutOfViewEvent, 50
 - StopBonusEvent, 65
- setHeight
 - Model::Entity, 32
- setMSort
 - Model::Bonus, 18
- Settings, 12
 - Difficulty, 13
- setWidth
 - Model::Entity, 33
- setWindowDimensions
 - Utils::Camera, 24
- setWorldDimensions
 - Utils::Camera, 24
- setX
 - Model::Entity, 33
- setY
 - Model::Entity, 33
- spawnBonus
 - World, 74
- spawnEntity
 - World, 74
- spawnPlatform
 - World, 74
- StopBonusEvent, 64
 - send, 65
- transform
 - Utils::Camera, 24
- Type
 - Model, 12
- Utils, 13
- Utils::Camera, 20
 - getInstance, 21
 - getMaxHeight, 21
 - getWindowDimensions, 21
 - getWorldDimensions, 22
 - getX, 22
 - getY, 22
- inverseTransform, 22
- isMaxHeight, 23
- move, 23
- operator=, 23
- setWindowDimensions, 24
- setWorldDimensions, 24
- transform, 24
- Utils::Exception, 35
- Utils::FileException, 36
- Utils::Random, 57
 - getInstance, 57
 - operator=, 57
 - random, 58
- Utils::Resourceholder< Type >, 58
- Utils::Resourcemanager, 59
- Utils::Stopwatch, 65
 - getDelta, 66
 - getInstance, 66
 - lap, 66
 - operator=, 67
- Utils::Utilities, 70
 - checkCollision, 70
 - checkWeight, 71
- View::BackgroundView, 16
 - handleEvent, 17
- View::BonusView, 19
- View::ConcreteFactory, 26
 - createBackground, 27
 - createHorizontalPlatform, 27
 - createJetpack, 27
 - createPlayer, 27
 - createScore, 27
 - createSpring, 27
 - createStaticPlatform, 28
 - createTemporaryPlatform, 28
 - createVerticalPlatform, 28
- View::IView, 41
 - handleEvent, 42
 - onTrigger, 42
- View::PlatformView, 52
 - handleEvent, 52
- View::PlayerView, 56
- View::ScoreView, 61
- visit
 - Model::Entity, 33
 - Model::Jetpack, 45
 - Model::Spring, 63
- Visitor::IVisitor, 43
- World, 72
 - addEntity, 73
 - events, 73
 - spawnBonus, 74
 - spawnEntity, 74
 - spawnPlatform, 74