

Klassen- und Methodenübersicht

Klasse Player (Oberklasse Object) Ein Spieler ist das Hauptelement im Spiel, du kannst ihn steuern und ihm Aufträge erteilen	
Konstr.	Player(double x, double y, double z, double xSize, double ySize, double zSize, String image) erstellt einen Spieler, an den Koordinaten „x“, „y“, „z“ mit der Größe „xSize“, „ySize“, „zSize“ und der Textur „image“
	Player(GLVektor position, GLVektor size, String image) erstellt einen Spieler, an den Koordinaten „position“ mit der Größe „size“ und der Textur „image“
	Player(GLVektor position, double size, String image) erstellt einen Spieler, an den Koordinaten „position“ mit der Größe „size“ und der Textur „image“
Anfrage	GLVektor getPosition() gibt die Position des Spielers
Anfrage	GLVektor getSize() gibt dir die Größe des Spielers
Anfrage	int getViewDirection() gibt dir die Blickrichtung des Spielers
Anfrage	double getSpeed() gibt dir die Geschwindigkeit des Spielers
Anfrage	double getGravitation() gibt dir die Gravitation des Spielers
Anfrage	int getJumpStrength() gibt dir die Sprungstärke des Spielers
Anfrage	boolean hitsBlock(GLVektor vektor) schaut ob der Spieler in der Richtung von „vektor“ einen Block berührt
Anfrage	ArrayList<GLVektor> getHitBox() gibt dir eine Liste von Koordinaten von der HitBox des Spielers
Anfrage	int getGameSpeed() gibt dir die Geschwindigkeit des Spiels
Auftrag	boolean move(GLVektor vektor) bewegt den Spieler in Richtung der Koordinaten „vektor“
Auftrag	void moveIgnoringHitBox(GLVektor vektor) bewegt den Spieler in Richtung der Koordinaten „vektor“, wobei er an anderen Blöcken nicht stehen bleibt
Auftrag	void setPosition(GLVektor vektor) setzt die Position des Spielers
Auftrag	void jump(int strength) lässt den Spieler um die Stärke „strength“ springen
Auftrag	void setJumpStrength(int jumpStrength) setzt die Sprungstärke des Spielers
Auftrag	void setSize(double xSize, double ySize, double zSize) setzt die Größe des Spielers
Auftrag	void setSpeed(double speed) setzt die Geschwindigkeit des Spielers
Auftrag	void setGravitation(double gravitation) setzt die Gravitation des Spielers
Auftrag	void invertPlayerGravity(double prozent) kehrt den Wert, mit dem der Spieler fällt um, sodass er statt fällt nach oben fliegt und anders herum. „prozent“ ist die Stärke der Umkehr Energie: 1.0 = 100% und 0.3 = 30%
Auftrag	boolean setKeyInputs(char[] inputs) setzt die Tasten, mit dem man den Spieler steuert. Gibt false, wenn die KeyInputs falsch waren. Der char-array muss aus 7 chars bestehen. Funktionen der chars: 1. nach links bewegen 2. nach rechts bewegen 3. nach vorne bewegen 4. nach hinten bewegen 5. springen 6. Kamera links rotieren 7. Kamera rechts rotieren
Auftrag	void setGameSpeed(int speed) setzt die Geschwindigkeit des Spiels auf „speed“ fps (frames per second)

Klasse Block (Oberklasse Object) Ein Block ist auch ein wichtiger Bestandteil des Spiels, er lässt dich Maps erstellen, auf welchen Blöcken du nun auch rumlaufen wirst. Außerdem bietet er noch die Möglichkeit als Überklasse, damit man verschiedene Arten von Blöcken programmieren kann, ohne dabei den ganzen Block noch einmal neu zu schreiben zu müssen	
Konstr.	Block(double x, double y, double z, double xSize, double ySize, double zSize, String image) erstellt einen Block, an den Koordinaten „x“, „y“, „z“ mit der Größe „xSize“, „ySize“, „zSize“ und der Textur „image“
	Block(GLVektor position, GLVektor size, String image) erstellt einen Block, an den Koordinaten „position“ mit der Größe „size“ und der Textur „image“
	Block(GLVektor position, double size, String image) erstellt einen Block, an den Koordinaten „position“ mit der Größe „size“ und der Textur „image“
Anfrage	GLVektor getPosition() gibt dir die Position des Blocks
Anfrage	GLVektor getSize() gibt dir die Größe des Blocks
Anfrage	GLVektor[] getHitPoints() gibt dir die Eckpunkte des Blocks
Anfrage	boolean hitsThisBlock(GLVektor vektor) schaut, ob die Position „vektor“ den Block berührt
Auftrag	boolean move(GLVektor vektor) bewegt den Block und gibt dir einen Wert zurück, je nachdem ob er sich bewegen konnte oder nicht
Auftrag	void setSize(double xSize, double ySize, double zSize) verändert die Größe des Blocks
Auftrag	void setPosition(GLVektor vektor) setzt die Position des Blocks
Auftrag	void setImage(String image) setzt die Textur des Blocks
Event	boolean onPlayerHitBlock(Player player, GLVektor vektor) diese Methode wird vom Player aufgerufen, falls er diesen Block berührt

Klasse Box (Oberklasse Block) eine Box ist eine Unterklasse vom Block, die es dir ermöglicht, verschiebbare Blöcke zu erstellen, welche auch von der Gravitation beeinflusst werden	
Konstr.	Box(double x, double y, double z, double xSize, double ySize, double zSize, String image) erstellt eine Box, an den Koordinaten „x“, „y“, „z“ mit der Größe „xSize“, „ySize“, „zSize“ und der Textur „image“
	Box(GLVektor position, GLVektor size, String image) erstellt eine Box, an den Koordinaten „position“ mit der Größe „size“ und der Textur „image“
	Box(GLVektor position, double size, String image) erstellt eine Box, an den Koordinaten „position“ mit der Größe „size“ und der Textur „image“
Anfrage	boolean hitsBlock(GLVektor vektor) gibt dir einen Wert zurück, ob der Block sich in die Richtung „vektor“ bewegen kann

Klasse Teleporter (Oberklasse Block) ein Teleporter ist eine Unterklasse vom Block, der es dir ermöglicht bei Berührung einen Spieler an eine bestimmte Stelle zu teleportieren	
Konstr.	Teleporter (double x, double y, double z, double xSize, double ySize, double zSize, String image, double spawnX, double spawnY, double spawnZ) erstellt einen Teleporter an den Koordinaten „x“, „y“, „z“ mit der Größe „xSize“, „ySize“, „zSize“ und der Textur „image“ und der SpawnLocation „spawnX“, „spawnY“, „spawnZ“
	Teleporter (GLVektor position, GLVektor size, String image, GLVektor spawn) erstellt einen Teleporter, an den Koordinaten „position“ mit der Größe „size“ und der Textur „image“ und der SpawnLocation „spawn“
Anfrage	GLVektor getTeleportLocation() gibt dir den Ort, an dem ein Spieler beim berühren teleportiert wird
Auftrag	void setTeleportLocation(GLVektor vektor) setze die TeleportLocation, an dem ein Spieler beim berühren teleportiert wird



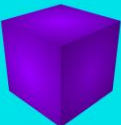
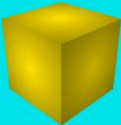
Klasse MapLoader(Oberklasse Teleporter) Der MapLoader ermöglicht während des Spielens eine Map zu laden	
Konstr.	MapLoader(double x, double y, double z, double xSize, double ySize, double zSize, Map map, String image, double spawnX, double spawnY, double spawnZ) erstellt einen MapLoader an den Koordinaten „x“, „y“, „z“ mit der Größe „xSize“, „ySize“, „zSize“ und der Textur „image“ und der SpawnLocation „spawnX“, „spawnY“, „spawnZ“
	MapLoader (GLVektor position, GLVektor size, Map map, String image, GLVektor spawn) erstellt einen MapLoader, an den Koordinaten „position“ mit der Größe „size“ und der Textur „image“ und der SpawnLocation „spawn“

Klasse JumpPad(Oberklasse Block) Das JumpPad, welches auch eine Unterklasse vom Block ist, ermöglicht es dir, den Spieler bei Berührung mit einer gewissen Stärke springen zu lassen	
Konstr.	JumpPad (double x, double y, double z, double xSize, double ySize, double zSize, String image) erstellt ein JumpPad, an den Koordinaten „x“, „y“, „z“ mit der Größe „xSize“, „ySize“, „zSize“ und der Textur „image“
	JumpPad (GLVektor position, GLVektor size, String image) erstellt ein JumpPad, an den Koordinaten „position“ mit der Größe „size“ und der Textur „image“
	JumpPad (GLVektor position, double size, String image) erstellt ein JumpPad, an den Koordinaten „position“ mit der Größe „size“ und der Textur „image“

Klasse Game(Oberklasse Object) Die Klasse Game beinhaltet einen BlockManager, und einen Integer, der die Spielgeschwindigkeit angibt	
Konstr.	Game(int gameSpeed) erstellt ein Spiel mit einer Spielgeschwindigkeit z.B. 60 = 60 Bilder pro Sekunde

Klasse Map (Oberklasse Object) Eine Map kann viele Blöcke beinhalten und dient dazu, dass man verschiedene Blöcke zu bestimmten Zeitpunkten laden kann	
Konstr.	Map() erstellt eine Map, die verschiedene Blöcke beinhalten kann
Anfrage	ArrayList<Block> getBlocks() gibt dir eine Liste von den Blöcken in der Map
Auftrag	void addBlock(Block block) fügt der Map einen Block hinzu

Klasse BlockManager(Oberklasse Object) Der BlockManager berechnet alle geladenen Blöcke, die gerade im Spiel sind	
Konstr.	BlockManager() erstellt einen BlockManager, wird aber nicht benötigt, da die Klasse Game dies bereits tut
Anfrage	ArrayList<Block> getBlocks() gibt dir eine Liste aller Blöcke
Anfrage	ArrayList<Block> getBlocks(GLVektor vektor) gibt dir eine Liste aller Blöcke an der Position „vektor“
Anfrage	boolean hitsBlock(GLVektor vektor) gibt dir einen Wert zurück, je nachdem ob an der Position ein Block ist oder nicht
Anfrage	boolean hitsBlock(GLVektor vektor, Block searchedBlock) gibt dir einen Wert zurück, je nachdem ob an der Position ein bestimmter Block ist oder nicht
Anfrage	boolean hitsOtherBlock(GLVektor vektor, Block executedBlock) gibt dir einen Wert zurück, je nachdem ob an der Position ein anderer Block, außer dem übergebenen ist oder nicht
Auftrag	void addBlock(Block block) füge einen Block dem Spiel hinzu
Auftrag	void removeBlock(Block block) entfernt einen Block aus dem Spiel
Auftrag	void loadMap(Map map) lädt alle Blöcke der Map in das Spiel

	<p>Block Ist ein normaler Block, auf welchem man sich bewegen und springen kann. Er wird mit einer weißen Textur gekennzeichnet.</p>
	<p>JumpPad Das JumpPad, eine Unterklasse der Klasse Block, dient dazu, dass bei Berührung eines Spielers dieser anfängt zu springen. Die Sprungstärke ist dabei von dem Block festgelegt, sodass es stärker oder schwächer als ein normaler Sprung sein kann</p>
	<p>Teleporter Der Teleporter ist eine Unterklasse der Klasse Block und besitzt Koordinaten, an denen ein Spieler bei Berührung teleportiert wird.</p>
	<p>MapLoader Der MapLoader ist eine Unterklasse der Klasse Teleporter, sodass der Spieler bei Berührung nicht nur zu einer bestimmten Position teleportiert wird, sondern auch eine neue Map geladen wird, die dem Objekt vorher zugewiesen wurde.</p>