TECHNICAL MANUAL

INTRODUCTION

Our project is an object-oriented implementation of the original Bomberman game with an additional multiplayer option via TCP.

USER STORIES

As a <char type>	I want to <do something>	so that I can <achieve an end result>	Notes
Player	play a game that runs smoothly	enjoy the game	
Player	use the arrow keys	move the bomber man around the map	
Player	play against other people online	test my skill against real people.	The network is local and the server is the game instance itself.
Player	know the number of monsters currently alive	know how many more I have to kill	
Player	choose between single player and multiplayer	chose how I want to play the game	
Player	be able to win or lose	have a purpose to play the game	
Online player	know how many people are playing with me	decide on my strategies	
Online player	limit the number of players to maximum 3 players	have a better chance of winning and have a balance game	
Online player	host a game online	challenge other people	
Online player	connect to a game hosted online	play against the challenger	
Online player	start the game at different locations on the map with my enemies	collect more power before meeting him	
Host	be the only person to start the game	have absolute authority whether or not to start the game.	
Client	enter the IP address of the host	connect to the host player	
Game object	know when I collide with another objects	react accordingly	

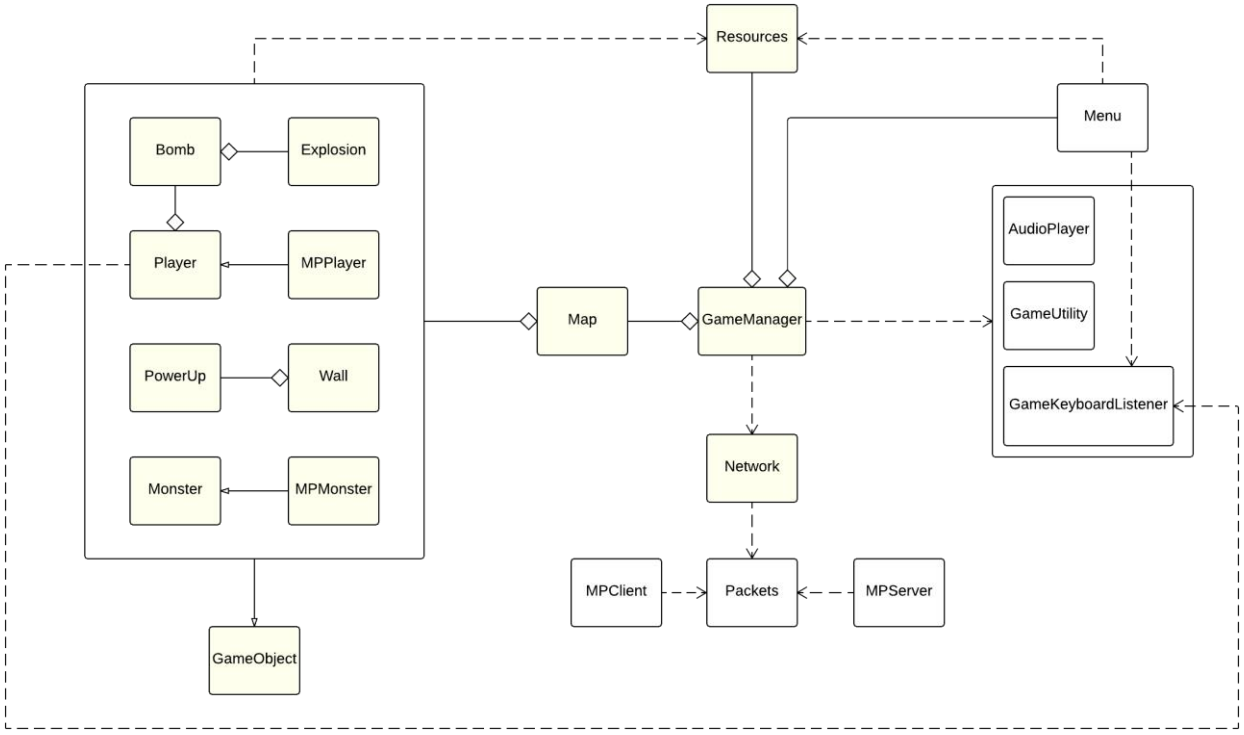
Bomber man	power up	be stronger and kill monsters faster and defeat other players	
Bomber man	plant bombs	kill monsters, destroy walls and defeat other players	
Bomber man	move around the map	find and defeat other monsters and players	
Monster	kill the player when I touch them	defeat the player	If have time, should create more than one kind of monster.
Monster	move around the map	kill the player easier	
Bomb	wait a short amount of time before explode	give the bomber man time to hide	
Bomb	explode	destroy other objects	
Wall	block the movement of other objects	limit their movements and make the game more challenging	
Wall	block the explosion	protect behind objects in the game	
Power up	be absorbed by the bomber man	power up the bomber man	
Power up	power up the bomber man in term of speed, bomb limit or explosion size	make the bomber man stronger	
Map	have walls randomly	limit monsters' and bomber man's movement.	
Map	spawn monsters occasionally	defeat the player easier	
Game designer	animate game objects	make the game look livelier and more interesting	
Game designer	have background music	boost the player's experience when playing the game	
Game designer	have sound effects	boost the player's experience when playing the game	

User stories chart

The game we created works in the same manner as the original Bomberman game. Players are able to control everything from their keyboard inputs. They can choose between "Normal Game" (the single player game), "Battle Mode" (the multiplayer game), or "Exit" in the main menu. In both types of game, players can control their characters' movements with arrow keys or WASD, and plant bombs using space bar. Players are created in the corners. They cannot walk through walls and bombs. Bombs can destroy everything that is destroyable, including walls, monsters and players, by creating explosions. Non-destroyable walls are created at fixed locations and destroyable walls are created randomly. Power-ups are items that can increase the players' attributes. They are randomly put in destroyable walls. Monsters walk

randomly around the map and can destroy players by walking through them. They spawn randomly at a fixed period and have the same collision mechanism as players'. Players can see the number of monsters on the sidebar. In the single player version, player has to destroy all of the monsters on the map in order to win. If the player choose to the multiplayer version, he or she will be taken to another screen asking if he or she wants to be the host of the game. If the player chooses to be the host, he or she will be taken to the waiting screen and can only start the game when there are more than 1 player. On the other hand, if the player chooses to be the client, he or she will be taken to a screen where he or she will have to enter the host's IP address. After that, he or she will also be at the waiting screen, but cannot start the game. In a multiplayer game, players have to destroy all of the other players in order to win. In addition, the game has background music and sound effects for placing bombs, bombs exploding, players getting destroyed, winning and losing.

OBJECT-ORIENTED DESIGN

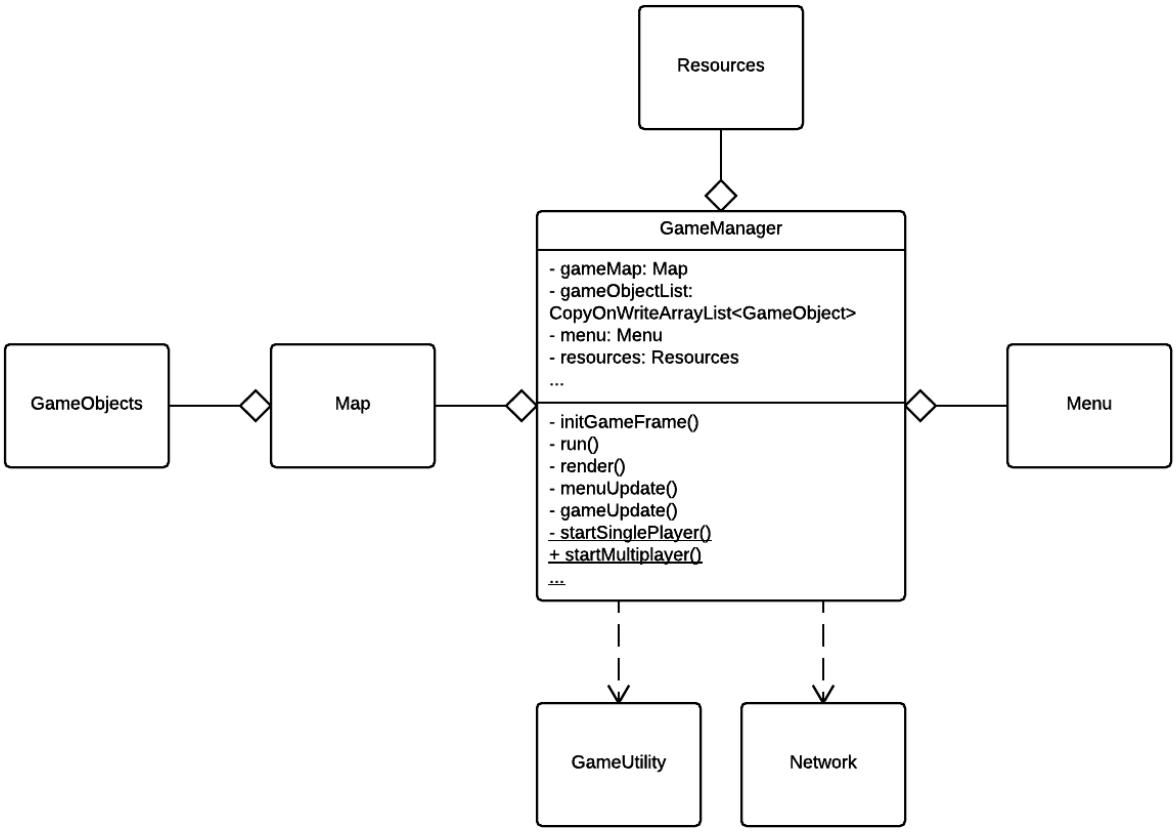


Basic UML diagram

GAMEMANGER

GameManager	
Responsibilities	Collaborators
• Create a frame • Create a loop to constantly update the game (game loop) • Switch menu screens • Create a map • Create resources • Switch between states (game, menu, win/lose) • Start game (single player or multiplayer) • Play background music • Receive and process info from network	• GameObjects (package) • Network (package) • GameUtility (package) • Menu

CRC card for GameManager



GameManager class diagram

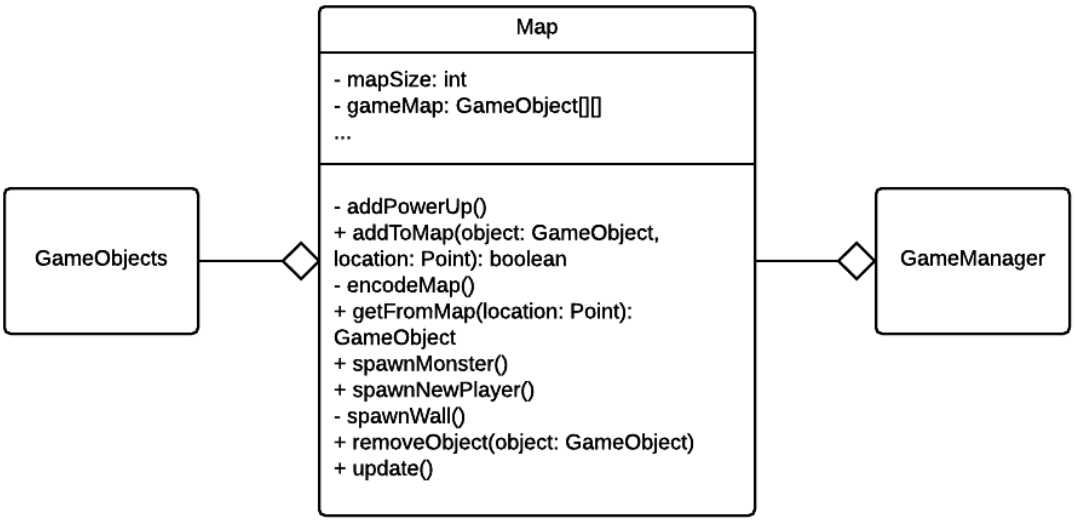
GameManager is the core of the program. It is responsible for creating (directly and indirectly) and updating the objects needed for the game (frame, game loop, map, resources), and receiving information from network. This class ensures that players have a smooth playing experience.

MAP

Map	
Responsibilities	Collaborators
• Create walls • Create players • Create monsters • Encode itself to send to network • Assign power-ups to walls • Add objects • Remove objects • Retrieve objects	• GameObjects (package) • Network (package) • GameUtility (package) • GameManager

CRC card for Map

Map stores objects in the grid-based map. The grid-based map is a 2D array in which each item is a "tile" of the map. Although Map is responsible for creating players and monsters, it only contains stationary objects such as walls and bombs. This is the players' "world".

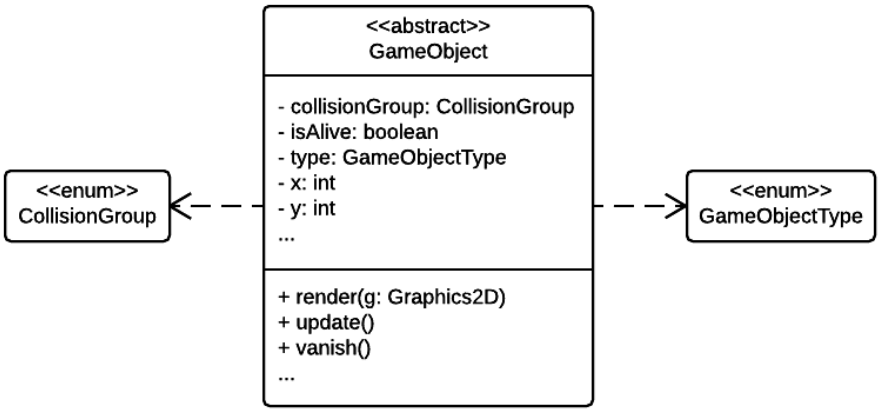


Map class diagram

GAMEOBJECT

GameObject	
Responsibilities	Collaborators
• Have a collision group • Have a location • Have a type • Update • Vanish	• Map • GameManager • Network

CRC card for GameObject



GameObject class diagram

GameObject is an abstract class that acts like a blueprint for all of the objects in the game. It contains attributes and methods that a game object must have. Collision group is used for distinguishing how an object should interact with other objects in order to react appropriately according to the rules of the original Bomberman game. For example, a player object cannot go through walls or bombs and can die from colliding with monsters or explosions.

PLAYER (MPPLAYER), BOMB & EXPLOSION

Player	
Responsibilities	Collaborators
• Move around • Plant bomb • Pick up power-ups • Store power attributes and an ID for multiplayer	• Map • Resources • Bomb • PowerUp

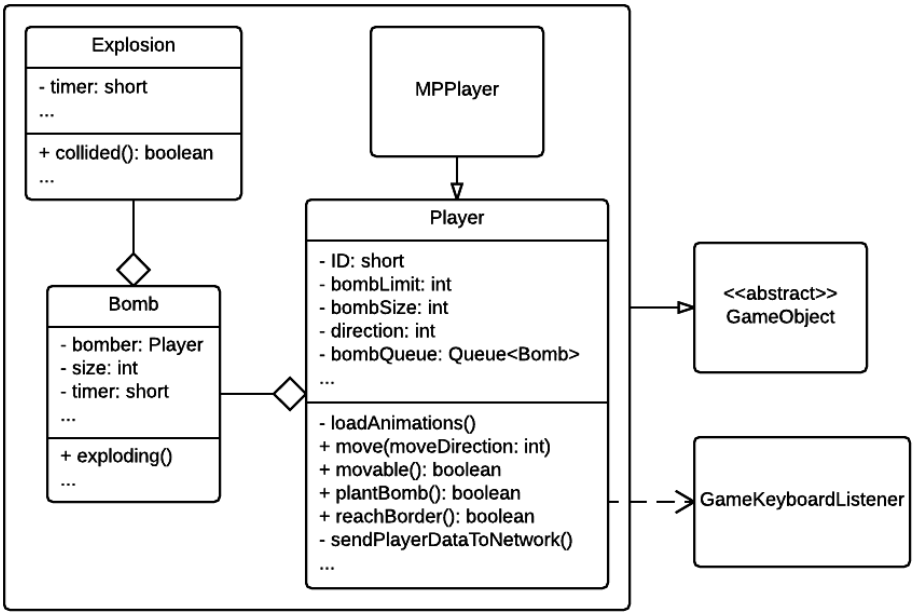
CRC card for Player

Bomb	
Responsibilities	Collaborators
• Create explosion • Vanish after exploding	• Player • Explosion • Resources

CRC card for Bomb

Explosion	
Responsibilities	Collaborators
• Destroy everything that is destroyable • Vanish	• Map • Resources • Bomb

CRC card for Explosion



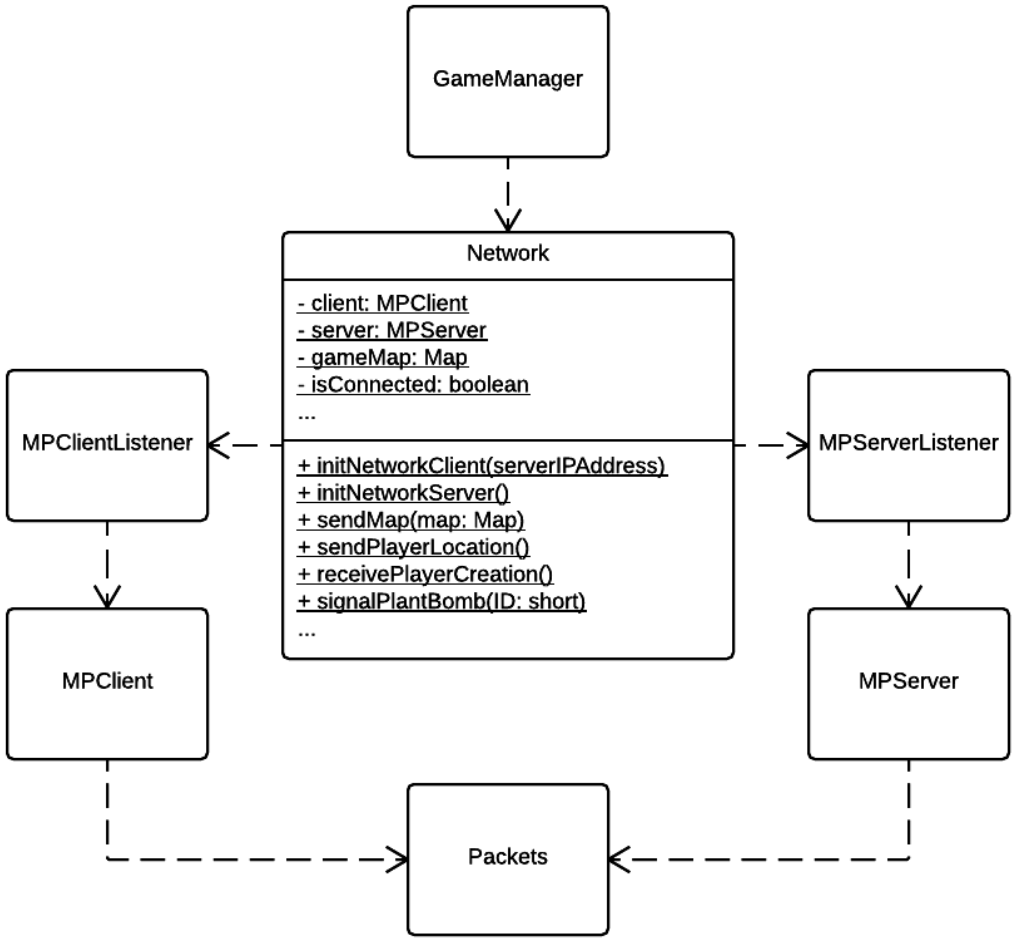
Class diagram

All of the above classes inherit directly from the GameObject class, except for MPPlayer. MPPlayer inherits from Player. It is an avatar of the real player controlled by another in the network. It receives data from network and acts accordingly. Player can walk around the map and plant bombs using keyboard inputs. Each Bomb object has a timer so it knows when to explode. When a Bomb object explodes, it creates an amount of Explosion objects equal to the bombSize instance of the Player who plants the bomb. Explosion objects also have a timer and a collided method to detect if they hit anything. These classes let the user have the basic experience of a Bomberman game.

NETWORK

Network	
Responsibilities	Collaborators
• Receive and store updated info from other games in the network	• MPClient • MPServer • GameManger

CRC card for Network



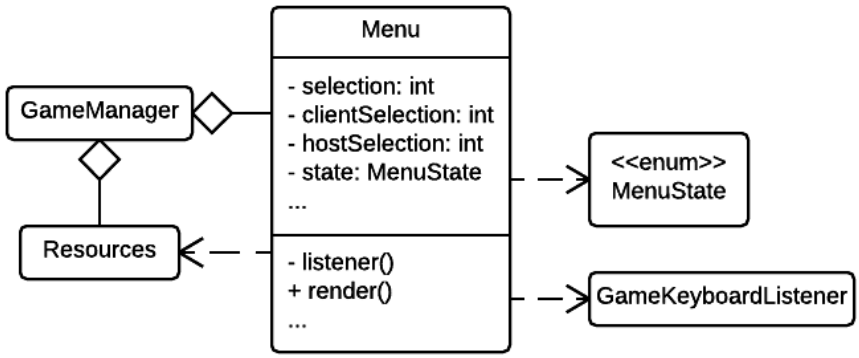
Network class diagram

We use a library called Kryonet to create out TCP network. MPClient and MPServer register and send packets to the server. Packets contain extremely small size information such as players' IDs, location, a string version of the game map, etc. After packets are sent, they will be processed by MPClientListener and MPServerListener so that the information received will be put in the right place in the Network class. The Network class is crucial for a multi-player game. It acts as the bridge between the main game and other games on the network by temporary storing the updates from other game instances, so that the GameManager class can access those updates and update the game accordingly.

MENU

Menu	
Responsibilities	Collaborators
• Let users choose between multiplayer and single player • Display winning/losing screen	• GameManager • GameKeyboardListener • Resources

CRC card for Menu



Menu class diagram

Menu is created so that players can choose between different modes of the game. It uses images loaded in a Resources object and updates itself using the GameKeyboardListener class. The Menu object has different states so that it knows which menu screens to display.

GAMEUTILITY

- *AudioPlayer:* This is for playback sound effects and looping the background music using Java Clip.
- *GameKeyboardListner:* This is for controlling the menu and the Bomberman.
- *GameUtility*: This is for calculating distances between two points, detecting collision, and converting between frame-based coordinates and grid-based coordinates.
- *Resources*: This is for storing images and sounds.

REFERENCES

http://www.kryonet.com/