

Bulletnom: The Game

Frames start with asterisks.

Id tag for server is always zero on instantiation. Id's for players are based on connection order to server: whenever a client connects, it gets the lowest available id, 1-3. If player two disconnects, for example, the next player to log on gets id 2.

Frame types

Bullets (Server to Client only): Short Bullet ID, Byte Type of bullet, Vector Vector, Short (x2) Starting Position

Enemies (Server to Client only): Short Enemy ID, Byte Type of enemy, Int Health, Vector Vector, Short (x2) Originating Position, Vector Acceleration (will be set to 0 for purposes of lab)

Requests to Move/Stop Moving (Client to Server only): Byte Direction (8 is up, 2 is down, 6 is right, 4 is left, 5 is stop)

CurrentLocation (Server to Client only): Short (x2) Coordinates (upon player move), Short Twist (how the turret is oriented in respect to 0 degrees)

Request to Fire (Client to Server only): 0 for stop, 1 for start (Byte)

Request to Turn (Client to Server only): 0 for stop, 4 for counterclockwise, 6 for clockwise (Byte)

Request Secondary Fire (Client to Server only): 0 for stop, 1 for start (Byte)

Request Bomb (Client to Server only): (Can be empty)

Ready (Client to Server only): (Can be empty)

ReadyStatus (Server to Client only): Byte Ready Status

Status (Server to Client only): Byte # of bombs, Short Energy, Long Score

KillPlayer (Server to Client only): (Can be empty)

KillGame (Server to Client only): (Can be empty)

Tick: Byte Clock

Request Refresh (Client to Server only): (Can be empty)

Refresh (Server to Client only): Location of everything (FreezeGame, send all bullet and enemy frames, send CurrentLocation of all players, send Status, wait for Ready)

PlayerDie (Server to Client only): (Can be empty)

EnemyDie (Server to Client only): Short Enemy ID, send status frame

FreezeGame (Server to Client only): (Can be empty)

ResumeGame (Server to Client only): (Can be empty)

PlayerRevive (Server to Client only): (Can be empty)

Respond to Tick (Client to Server only): (Can be empty)

Login (Both ways): String ID (Client to Server), (Can be empty) (Server to Client)

Logout (Server to Client only): (Can be empty)

SERVER TO CLIENT

0 Tick

1 Status

2 ReadyStatus

3 CurrentLocation

4 Bullets

5 Enemies

6 PlayerDie

7 EnemyDie

8 BulletDie

9 PlayerRevive

10 FreezeGame

11 ResumeGame

12 Refresh

13 KillPlayer

14 KillGame

CLIENT TO SERVER

15 RespondToTick

16 Ready

17 RequestMove

18 RequestTurn

19 RequestFire

20 RequestSecondaryFire

21 RequestBomb

22 RequestRefresh

23 Logout

BOTH

24 Login

Sample Frame

***208**

* marks the beginning of the frame

2 is the ID of the frame (this frame comes from Player 2, therefore the ID is 2)

0 is the number of bytes after the frame type (the ready frame is always empty)

8 is the frame type (this is a ready frame, therefore it is of type 8)

Protocol

Tick: Server sends the tick. This is sent every $1/60^{\text{th}}$ of a second and holds top priority. It expects a RespondToTick from all clients, with the correct tick. If the ticks do not match, the client sends a RequestRefresh and the server sends a FreezeGame to pause gameplay until laggy client gets caught up. Because the tick is one byte, it will loop when the tick reaches 255. There is the potential that the client is off by exactly 256 ticks, but it is highly unlikely that it will persist at that margin.

Status: Server sends the status when the game starts up and after each enemy kill. This contains the # of bombs, the energy, and the score of the team.

ReadyStatus: At the beginning of the game and after every FreezeGame, this is sent to ask for a Ready from each client to ensure that the game does not start when the players are not.

CurrentLocation: This gets sent whenever any client sends a RequestMove and the move is resolved by the server.

Bullets: These are sent whenever the server instantiates a bullet, usually when an enemy or ally shoots. The type determines whether or not it will hit enemies or allies.

Enemies: These are sent whenever the server instantiates an enemy, usually when they are spawned in some pre-determined pattern. The enemy for the purposes of this lab will only move in a vector, unmodified by a 0 magnitude acceleration. They have no turns, although an enemy that turns will simply destroy the enemy without granting points or energy and will spawn a new enemy with a new vector (and possibly a new acceleration).

PlayerDie: This is sent when the player is hit (a player is automatically killed when it gets hit). There is no information within the frame – it simply tells the client that the player is dead. Clients may still control their character (as a ghost), but they cannot fire nor receive damage/die again.

EnemyDie: This is sent when an enemy's HP reaches 0 or below. The enemy does not exist on the server anymore. This also grants the players 60 energy and some points, so this frame also sends a status frame.

BulletDie: This is sent when a bullet hits a solid object. The bullet does not exist on the server anymore. It deals damage (unless if it collides with the defender's shield or an invulnerable object) before dying.

PlayerRevive: When a player uses a bomb and at least one player is dead, the server sends this frame to a client (determined in the following order: defender (client 1), machine gunner (client 2), sniper (client 3)). A revived player will be alive and will be invulnerable to damage for .5 seconds.

FreezeGame: When a RequestRefresh is sent, a Refresh is sent, a FreezeGame occurs and the entire game simply pauses all actions (all objects have 0 velocity and cannot fire). The FreezeGame then expects all clients to respond with a Ready frame.

ResumeGame: When all players have responded with a Ready frame, a ResumeGame frame is sent once the action resumes (all velocities are resumed and all objects can fire again).

Refresh: When a RequestRefresh is received, the server sends a FreezeGame and it sends all current enemy and bullet frames. It then sends the CurrentLocation of all players and finally sends a Status frame to the client. The server then expects a Ready frame from all players.

KillPlayer: Kicks the player from the server (and closes the socket).

KillGame: KillPlayer all players from the server and exits the application.

RespondToTick: When the client receives a Tick from the server, it sends a RespondToTick. This frame has highest priority. This frame basically says “everything is normal.” The client-side method determines whether or not it is matched up to the server (it is not the server’s consequence if the client lags) and RequestRefresh if it is off.

Ready: The client sends this at the beginning of the game and after a FreezeGame. It cannot send this when it is receiving the payload of a Refresh frame. This essentially is the client saying “yes, I’m good, run the game.”

RequestMove: The client requests to move its ship/avatar. The byte number (direction) is determined by the direction on a numpad (8 being up, 4 being left, 6 being right, 2 being down, 5 being “brake” or remain still). This is sent whenever the directional buttons are pushed down and whenever they are released. The server then sends a CurrentLocation.

RequestTurn: The only client that may do this is client 3 (the sniper). This controls the turret on a sniper and the frame is sent when the turn button is pushed down and whenever it is released. The server then sends a CurrentLocation.

RequestFire: The client sends this whenever the fire key is pressed down and whenever it is released (byte is determined by: 1 to fire, 0 to stop).

RequestSecondaryFire: This operates like the RequestFire, but it applies to the defender’s shield.

RequestBomb: This is sent when the client presses a bomb button (there is a 5 second delay before the server recognizes another one of these frames from the same client). The server also sends a PlayerRevive if the conditions of PlayerRevive are fulfilled.

RequestRefresh: When the client recognizes that it is not synchronized with the server in ticks, it sends this frame.

Logout: When the client clicks the logout button, this is sent. Effectively, the socket is closed and a new socket is instantiated. The server now sends a Login frame.

Login: The server first sends an empty frame asking for login information of the client. The client then sends their ID and are logged onto the server.

Other Special Information

- Login
 - o Client send a login Frame
 - o Server accepts creates instance of game and replies with a login Frame
 - o Server opens Lobby and waits for other clients to join
 - o After 3 clients have gathered, they wait until everyone has sent a ready frame
Game begins

- Game Play
 - o Virtual world is created
 - o Clients send requests to server to do anything via various request frames
 - o Server sends the location of players, creation of enemies and bullets as well as deaths of anything along with status updates and revives of characters
 - o If a Client falls behind on their ticks then they will send a Request Refresh frame. This will cause the server to send a Freeze frame to stop the game for all players. Who will wait until all players send ready frames. The Client that requested the refresh will wait until they have been updated with all the current locations of bullets, enemies, and allies.
 - o When an enemy is killed the server updates the score and send an updated status frame
 - o If a player is shot they will die. They will still be able to see their ship and move around, but they will be unable to shoot or block bullets. When a player *that is alive* uses a bomb the players that were dead will be revived along with the effects of the bomb. The newly revived players will be invincible for .5 seconds and then return to normal.
 - o When all players are killed the server will end their instance of the game and return them to the lobby.