1. ¿Cómo jugar?

Para jugar podremos utilizar dos alternativas, el control por teclado o el control táctil.

- Si utilizamos el control por teclado, utilizaremos las flechas direccionales izquierda y derecha para desplazarnos, y el botón espacio para realizar una acción
- Si utilizamos el control táctil, utilizaremos el pad táctil para desplazarnos, y el botón de acción para realizar la acción.

Cuando hablamos de acción, solo podrá haber dos posibles:

- Al comenzar un nivel, utilizaremos la acción para lanzar la bola y así comenzar a jugar.
- Si disponemos de la mejora pegajosa, y se nos pega una o varias bolas, podremos utilizar la acción para despegarlas.

Además, debemos tener en cuenta que, aunque los rebotes con los bloques harán un rebote directo en cuanto a direccionalidad de la bola, esto no se aplica al jugador. SI golpeamos a la bola con uno de los extremos, se notará como se modificará la velocidad horizontal de la bola.

2. ¿Cómo se gana y cómo se pierde?

En este juego, iremos avanzando por niveles a medida que **todos los bloques normales del nivel queden destruidos**. Destruir bloques y obtener mejoras nos conseguirá puntos. Si conseguimos superar una fase, nos llevaremos los puntos obtenidos a la siguiente fase.

Para perder, tendremos que quedarnos **sin bolas con las que jugar**. Esto nos dejará volver a comenzar el nivel, pero tendremos puntuación cero.

3. Tipos de mejoras

Al romper bloques normales habrá un porcentaje de probabilidad de que el bloque suelte una mejora (siempre pasará con los bloques de mejora). Estas mejoras son acumulables y se podrá beneficiar del efecto de varias a la vez.
En el juego habrá un total de cinco mejoras:

- **Mejora de velocidad**: Esta mejora se verá representada por el color azul y una flecha hacia la derecha. Al recogerla, el jugador verá incrementada su velocidad por un tiempo limitado.

- **Mejora de tamaño**: Esta mejora se verá representada por el color rosa y una flecha hacia la izquierda y la derecha. Al recogerla, el jugador verá incrementado su tamaño durante un tiempo limitado. También servirá para estabilizar la bola ya que se perderá el efecto del cambio horizontal de velocidad mientras dure el efecto.

- **Mejora x3**: Esta mejora se verá representada por el color morado y tres bolas blanca. Al recogerla, cada bola en juego se verá multiplicada por 3, con las mismas propiedades que el resto de bolas.

- **Mejora pegajosa**: Esta mejora se verá representada por el color verde y una mancha de pintura blanca. Al recogerla, el jugador cambiará su textura por una verde, y cuando las bolas boten contra el jugador se quedarán pegadas, siguiendo la dirección del jugador. Una vez el jugador realiza la acción, proseguirán su curso. Este efecto será temporal.

- **Mejora de fuego:** Esta mejora se verá representada por el color rojo y una llama. Al recogerla, las bolas se encenderán, cambiando su textura por una

roja. Las bolas que están encendidas ignorarán la vida de los bloques y los romperán al contacto sin rebotar. Este efecto será temporal.

4. Tipos de bloques

Hay varios tipos de bloques, aunque el único requisito para superar los niveles sea destruir todos los bloques normales y reforzados (si hay). Todos los bloques sumarán puntos al ser golpeados, siendo diferentes por cada bloque.

- **Bloques normales**: Este bloque podrá ser de color amarillo, azul, verde o rojo, con colores planos. Se destruirá al primer contacto con la bola, y será necesario destruir todos los presentes en el nivel para avanzar. Una vez destruidos, tendrán cierto porcentaje de soltar una mejora aleatoria.

- **Bloques reforzados**: Estos bloques son de un azul claro con tonalidades diferentes a los normales. Además, serán necesarios dos golpes para que se destruyen y al ser considerados como bloques destruibles, será necesario destruirlos todos también para completar el nivel. También hay cierta probabilidad de que suelten mejora.

- **Bloques mejora**: Estos bloques también necesitarán dos golpes para ser destruidos, pero siempre otorgarán una mejora.

- **Bloques bomba**: Estos bloques crearán una explosión al ser golpeados, que golpeará a los bloques adyacentes, rompiéndolos.

- **Bloque eliminatorio:** Estos bloques no se pueden romper, si no que eliminarán a la bola si entra en contacto. Si la única bola en juego entra en contacto con este bloque, se pierde la partida.

5. Niveles

Los primeros cinco niveles serán niveles para probar el funcionamiento de cada bloque. Los restantes serán niveles diseñados para el juego.

- **Nivel 0:** Un nivel con unos pocos bloques normales

- **Nivel 1**: Un nivel con unos pocos bloques normales y un par de bloques mejora

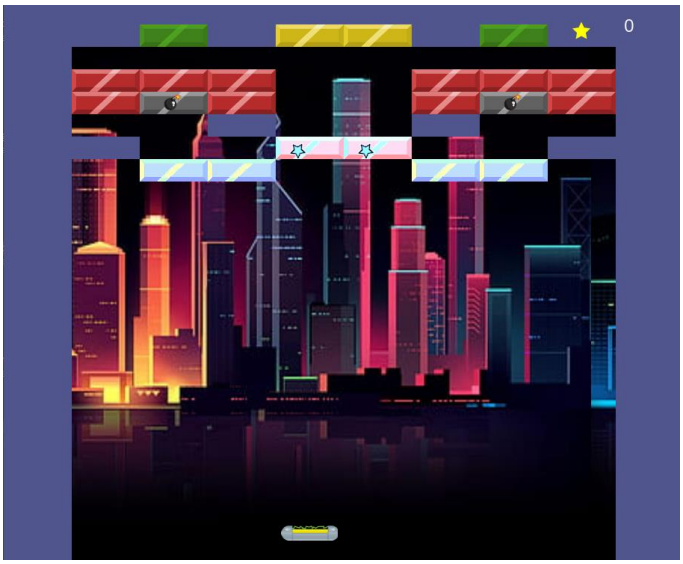
- **Nivel 2**: Un nivel con unos pocos bloques normales y un par de bloques bomba

- **Nivel 3**: Un nivel con unos pocos bloques normales y un par de bloques reforzados

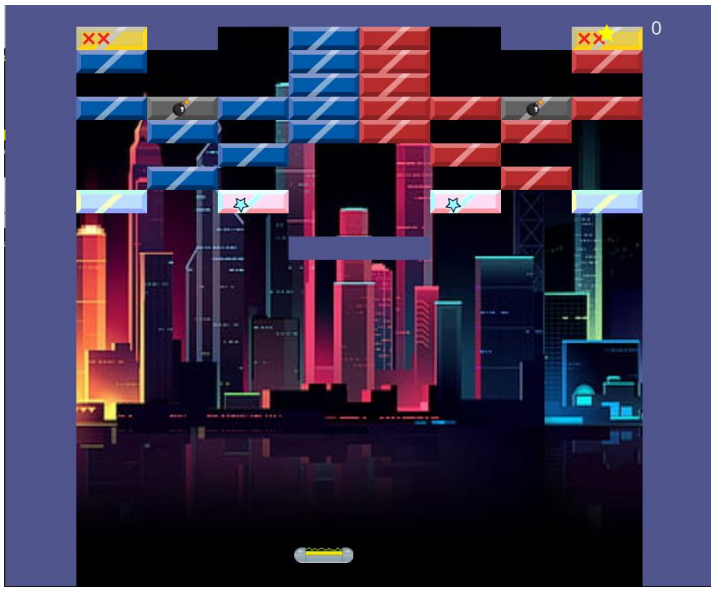
- **Nivel 4:** Un nivel con unos pocos bloques normales y un par de bloques eliminatorios

- **Nivel 5**: Nivel con solo bloques normales pero con una forma diferente de nivel para cuidar los rebotes.

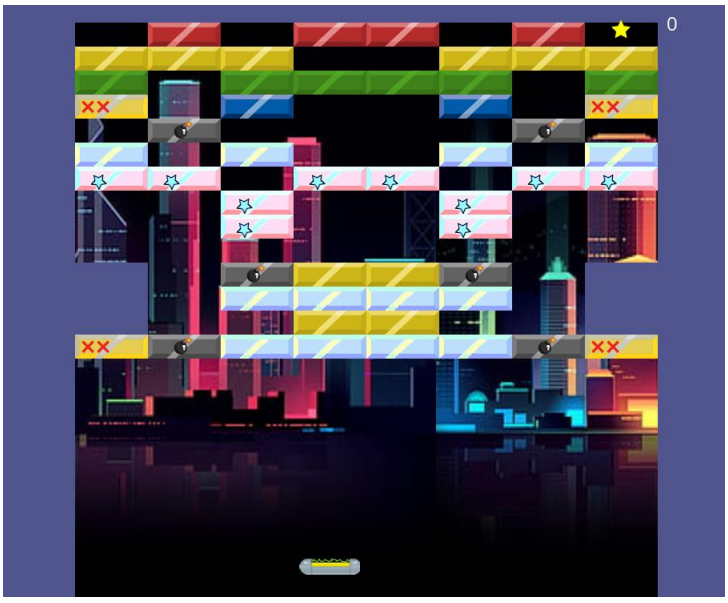
- **Nivel 6**: Nivel con solo bloques normales incrustados en los límites, además de un par de bloques mejora, reforzados y bloques bomba preparados para explotar varios bloques. También hay bloques de nivel de por medio para rebotes diferentes.



- **Nivel 7**: Nivel con la misma estructura a la izquierda que a la derecha. Bloques bomba para destruir un plar de bloques y un par de bloques eliminatorios.



- **Nivel 8**: El nivel más complicado. Gran cantidad de bloque y bloques bomba que facilitan varias estrategias.



6. Implementación

Voy a destacar lo que me pareció más importante/complicado/extraño.

Primero empezando por la **colisión de la bola con el jugador**, que cambiará la velocidad horizontal con la que saldrá despedida la bola dependiendo de la posición donde colisione con el jugador

```
for (auto const& bola : bolas) {
	if (bola->isAbove(player)) {
		if (player->pegajoso) {
			if (!bola->glued) {
```

```
                                audioStick->play();
                        }
                        bola->pegar();

                }
                else {
                        //audioBounce->play();
                }
                bola->vy = -bola->vy;
                if (player->timeSizeUpgrade <= 0) {
                        int bolacentro = bola->width / 2 + bola->x;
                        int playerWidth = player->width + player->x;
                        int playerwidthMIN = player->x - (player->width / 2);
                        float velocityBounceMin = (playerwidthMIN -
playerWidth) / 5 + 10;
                        for (int i = playerwidthMIN - 20; i < playerWidth +
20; i++) {
                                if (i == bolacentro) {
                                        bola->vx = velocityBounceMin;
                                }
                                if (i % 5 == 0) {
                                        velocityBounceMin++;
                                }
                        }
                }
        }
}
```

Ignorando la parte de pegajosidad que veremos luego, comprobamos que la bola está overlapeando pero encima del jugador (con el isAbove) y esto implica que va a rebotar por lo que sabemos que la velocidad vertical se invertirá. Para la horizontal, primero comprobamos que no está la mejora de tamaño, calculamos la superficie total del jugador y ajustamos con el for loop tal que dependiendo de en que x esté el centro del círculo, más o menos velocidad horizontal cogerá.

Ya que ha salido la **pegajosidad** la comentaremos. El jugador tendrá el estado pegajoso (boolean) que cambiará una vez recoja la mejora o se le acaba el tiempo. Las bolas en sí, tienen el estado glued, que se pondrá a true si justo colisionan con el jugador pegajoso. En el update de bola tenemos que:

```
void Bola::update() {
        if (!glued) {
                x = x + vx;
                y = y + vy;
                }

        ...
```

Así cuando esté pegada, no se moverá con movimiento natural. Sin embargo tendrá que acompañar al jugador. Esto se hará en Gamelayer, cuando miramos los controles.

```
// Eje X
        if (controlMoveX > 0) {
                for (auto const& bola : bolas) {
                        if (bola->glued) {
                                bola->x += player->speed;
                        }
                }
                player->moveX(1);
```

Para despegarla, nos fijamos en la acción:

```
if (controlShoot) {
		if (firstShoot) {
			Bola* newProjectile = player->shoot();
			if (newProjectile != NULL) {
				bolas.push_back(newProjectile);
			}
			firstShoot = false;
		}
		else {
			for (auto const& bola : bolas) {
				if (bola->glued) {
					bola->despegar();
					bola->y-=5;
				}
			}
		}

		}
```

Ajustamos algo la posición de la bola para que el despegar no vuelva a dar colisiones.

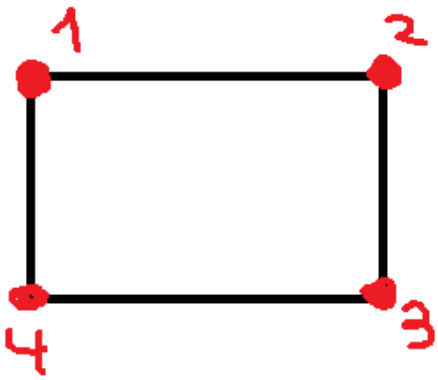
Cambiando a **colisiones con los tiles**, tendremos el método que recogerá la bola, el tile y si el bloque es destruible (para que no rompa tiles extremos de nivel)

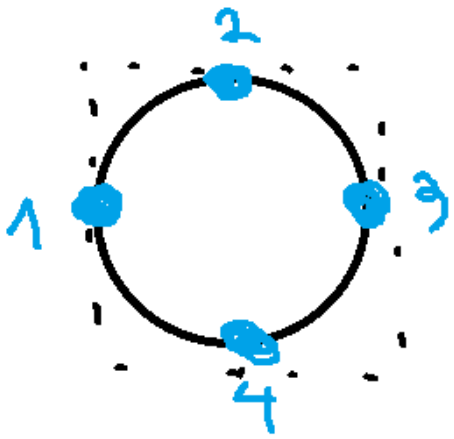
```
bool GameLayer::overlapBolaTile(Bola* bola, Tile* tile, bool destruible)
{
	int hotfix = -30;
	bool value = false;
	int tileX = tile->x;
	int tileWidth = tile->width;
	int tileY = tile->y;
	int tileHeight = tile->height;
	int bolacentroX = bola->width / 2 + bola->x;
	int bolacentroY = bola->height / 2 + bola->y;

	int tile1X = tile->x  + hotfix;
	int tile1Y = tile->y;

	int tile2X = tile->x + tile->width + hotfix;
	int tile2Y = tile->y;

	int tile3X = tile->x + tile->width + hotfix;
	int tile3Y = tile->y + tile->height;

	int tile4X = tile->x + hotfix;
	int tile4Y = tile->y + tile->height;

	int bola1X = bola->x;
	int bola1Y = bola->y + (bola->height / 2);

	int bola2X = bola->x + (bola->width / 2);
	int bola2Y = bola->y;

	int bola3X = bola->x + bola->width;
	int bola3Y = bola->y + (bola->height / 2);

	int bola4X = bola->x + (bola->width / 2);
	int bola4Y = bola->y + bola->height;
```

```
	//Choques verticales
	if (bola2Y >= tile1Y && bola2Y <= tile3Y) { //b2 verticalmente dentro del
tile
		if (bola2X >= tile1X && bola2X <= tile2X) {//b2 horizontalmente
dentro del tile
			if (!(bola->prendida && destruible)) {
				bola->verticalBounce();

			}
			value = true;
		}
	}

	if (bola4Y >= tile3Y && bola4Y <= tile2Y) { //b4 verticalmente dentro del
tile
		if (bola4X >= tile1X && bola4X <= tile2X) {//b4 horizontalmente
dentro del tile
			if (!(bola->prendida && destruible)) {
				bola->verticalBounce();

			}
			value = true;
		}
	}

	if (bola1X <= tile2X && bola1X >= tile1X) { //b1 horizontalmente dentro
del tile
		if (bola1Y >= tile1Y && bola1Y <= tile3Y) { //b1 verticalmente
dentro del tile
			if (!(bola->prendida && destruible)) {
				bola->horizontalBounce();

			}
			value = true;
		}
	}

	if (tile1X <= bola3X && tile2X >= bola3X) { //b1 horizontalmente dentro
del tile
		if (bola1Y >= tile1Y && bola1Y <= tile3Y) { //b1 verticalmente
dentro del tile
			if (!(bola->prendida && destruible)) {
				bola->horizontalBounce();

			}
			value = true;
		}
	}

	if (value && !bola->prendida) {
		//audioBounce->play();
	}
	return value;

	}
```

Antes de comentar los valores usados, cabe mencionar que comprobamos si la bola tiene la mejora de fuego para no rebotar, y devolveremos true si significa que hay colisión.

Entrando otra vez con la colisión, ya que el el método isOverlap de actor no podía determinar si el rebote será vertical o horizontal, decidí crear uno propio, cogiendo los extremos de los vértices del rectángulo. Para el círculo en vez de coger los vértices del cuadrado que contiene al círculo, escogí utilizar los puntos donde el círculo contenido y el cuadrado que lo contiene convergen, para que así las colisiones fueran más exactas.





Por último, comentaré el tema **bloques bomba y explosiones**. Funciona igual que un bloque normal hasta que colisiona con la bola, que generará la explosión.

```
for (auto const& tile : bombas) {
		for (auto const& bola : bolas) {
			if (overlapBolaTile(bola, tile, true)) {
				points += 20;
				if (tile->tocadoYHundido()) {
					points += 20;
					explosiones.push_back(new Explosion(tile->x,
tile->y, game));
					deleteTiles.push_back(tile);
				}
				textPoints->content = to_string(points);
			}
		}
	}
```

Es buen momento también para decir que el método tocadoYHundido resta uno de vida al tile, y si con eso se destruirá, devuelve true.

Aquí al destruirse, se ve que se genera una explosión que se meterá en un listado de explosiones que se updateará, hará draw etc.

```
for (auto const& explosion : explosiones) {
		explosion->update();
		for (auto const& tile : tilesDestructibles) {
			if (explosion->isOverlap(tile)) {
				points += 10;
```

```
                        if (tile->tocadoYHundido()) {
                                points += 10;
                                generarMejora(tile->x, tile->y, false);
                                deleteTiles.push_back(tile);
                        }
                        textPoints->content = to_string(points);
                }
        }

....
```

Luego será tan fácil como comprobar si las explosiones colisionan con un tile y lo destruiremos. Para que la explosión no se quede constantemente ahí, le metemos un lifespan que lo iremos restando cuando sea 0, y cuando sea 0, lo meteremos en un listado a borrar.

```
void Explosion::update() {
        lifespan--;
        }
```

8. Consideraciones importantes

El audio del rebote con los bloques y el pad está deshabilitado por no llenar de ruido la aplicación, aunque la línea se encuentra comentada en el código y el audio está metido en recursos (audioBounce en código, bounce en la carpeta res)