RELAZIONE PROGETTO OOP

- Badioli Filippo
- Francesco Petrassi
- Giovanni Minoccari

Alma Mater Studiorum - Università di Bologna
Corso di Programmazione ad Oggetti
Giugno 2023

INDICE

CAPITOLO 1

ANALISI

1.1 REQUISITI

Il software "TowerDefenseGame" è un gioco 2D del genere tower defense, sulla falsa riga di "Age Of War 2".
Lo scopo principale del gioco è quello di bloccare l'avanzata di una schiera di nemici, i quali non devono arrivare fino alla propria base, tramite lo schieramento di personaggi alleati.

Requisiti funzionali:

-Il giocatore dovrà essere in grado di generare degli alleati di diverse categorie che difenderanno la propria base dagli aggressori

-I nemici avanzeranno fino a che non incontreranno un alleato, a quel punto quello dei due che perde despawnerà, mentre l'altro perderà dei punti salute

-Il gioco progredisce ad ondate sempre più numerose e potenti

-Si guadagnano monete necessarie allo spawn degli alleati sia con il passare del tempo sia con l'uccisione dei nemici

-Se un nemico riesce ad arrivare fino alla base essa inizierà a perdere vita finchè un alleato non sconfiggerà il nemico in questione

-Se i punti salute della base scendono a zero la partita finisce

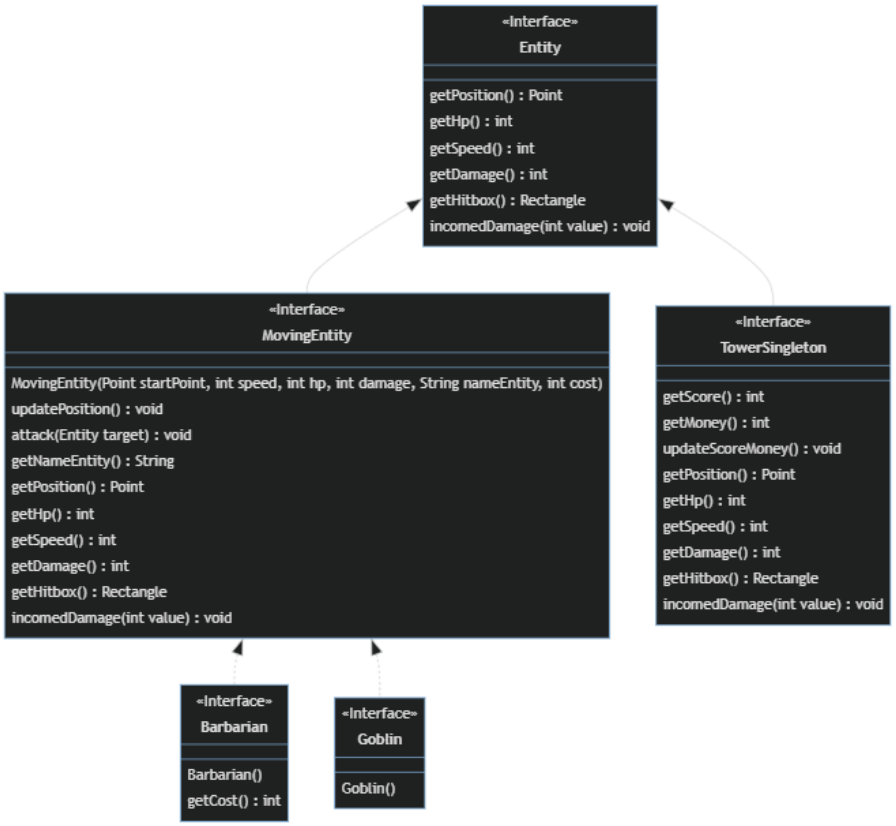
-Sarà possibile attivare una funzionalità speciale molto potente ma con un alto tempo di ricarica

-Una volta finita la partita all'utente sarà possibile effettuare un salvataggio del proprio punteggio

-Una classifica dei punteggi migliori può essere consultata dal menù iniziale

1.2 ANALISI E MODELLO DEL DOMINIO

L'entità TowerSingleton ricoprirà il ruolo principale rappresentando il giocatore e gestendo lo spawn delle MovingEntity.
Esse saranno, come deducibile dal nome, delle entità mobili con statistiche predeterminate(attacco, velocità, vita e costo) e si divideranno in due tipi, amiche e nemiche, diverse tra loro.
Ognuna delle "MovingEntity" avrà determinate differenze volte a caratterizzarle, come ad esempio, una maggiore vita per il Barbarian ma una velocità bassa.
Ci sarà infine un'ultima entità amica, ovvero la Turret, che sarà fissa sopra la TowerSingleton e, al pari delle MovingEntity, presenterà un attacco, una vita, un costo e una velocità pari a zero.
Le entità amiche e nemiche spawneranno ai lati opposti dello schermo e una volta entrate nel range di attacco, si scontreranno vicendevolmente, provocando danni fino alla morte di una delle due.

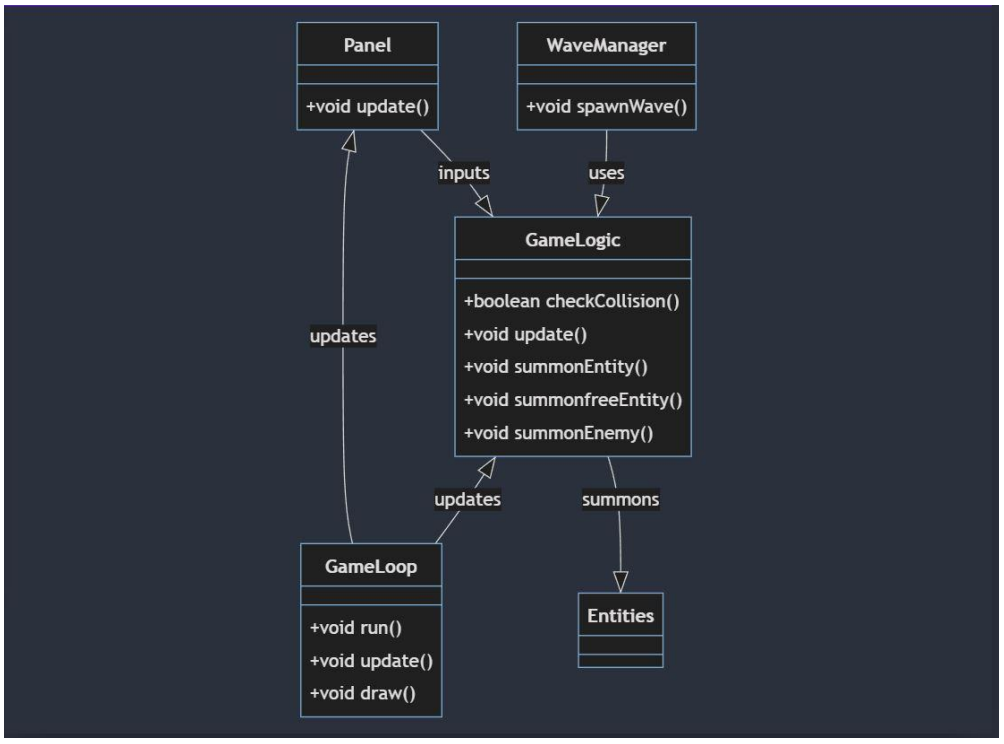


CAPITOLO 2

DESIGN

2.1 ARCHITETTURA

Come architettura per il software è stato scelto l'MVC.
La parte della "view" è affidata all'interfaccia Panel, responsabile di mostrare a schermo tutti gli avvenimenti e di catturare gli input dell'utente.
Il model invece è affidato al'interfaccia GameLogic, la quale si occupa di gestire la logica di base del programma e le varie entità.
Il gameloop invece è la sezione di programma che si occupa di aggiornare ciclicamente tutte le altre parti ed è la principale connessione tra view e model.
Se si dovesse cambiare in blocco la libreria grafica l'unica ripercussione sul programma sarebbe di dover cambiare i riferimenti ai JPanel presenti in altre classi/interfacce, mentre per inserire nuovi "potenziamenti" non ci sarebbero problemi di nessun tipo.
Per aggiungere delle entità (in particolare per i nemici) invece andrebbe modificato l'algoritmo dello spawnmanager
Un semplice UML che descrive queste relazioni:



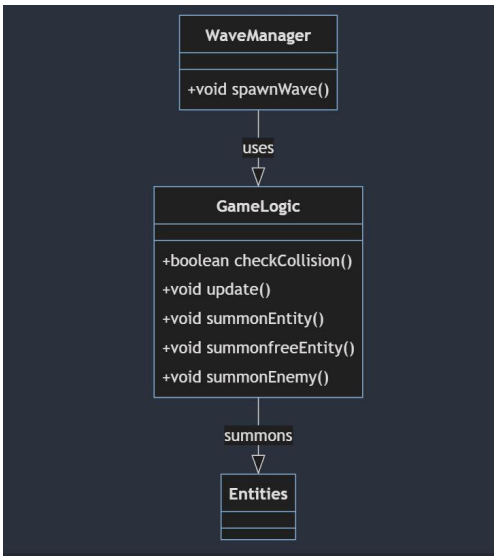
2.2 DESIGN DETTAGLIATO

BADIOLI

Spawn di entità nemiche

Problema: il software richiede lo spawn di ondate di nemici con difficoltà incrementale e presenza di diverse entità

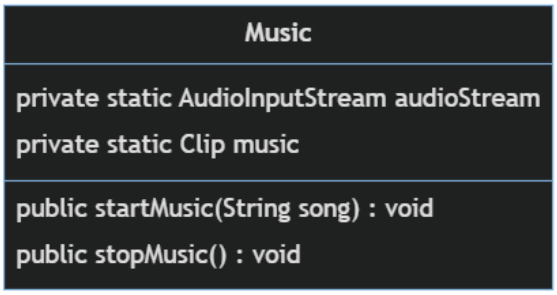
Soluzione: La gestione dello spawn è realizzata attraverso il pattern Singleton della classe WaveManager, creando una classe che si occupa di gestire internamente la composizione e la distanza temporale tra ondate di nemici.

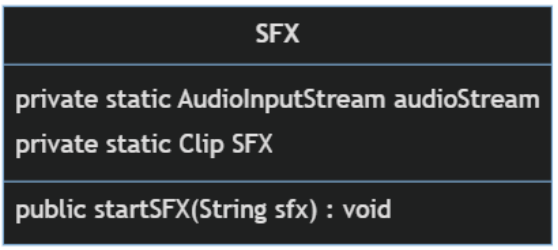


PETRASSI

Da suddivisione del progetto, stabilito dopo una discussione iniziale approfondita con i miei colleghi su come dividerci in maniera quanto più equidistribuita il progetto in questione, riporto quanto emerso in termini di risultati da me ottenuti e di problematicità riscontrate.

Problema: Come gestire la musica nei vari pannelli? E gli effetti sonori? Questi problemi nascono dal fatto che ogni pannello presente deve avere una propria musica di accompagnamento che deve accompagnare il giocatore nelle varie fasi del gioco, oltre che per marcare la differenza tra le varie fasi di gioco. Troviamo, inoltre, anche gli effetti sonori che devono essere riprodotti in contemporanea alla musica senza interromperla o sovrastarla.





Soluzione proposta: Inseriamo una classe Music che gestisce la musica in modo tale che ogni pannello quando viene creato, inserendo la stringa che fa riferimento al pannello nel metodo startMusic, può far partire la canzone corrispondente. E nel momento in cui viene generato un altro panello, dello stesso tipo o diverso, è possibile interrompere la musica, affinché il pannello successivo possa riprodurre la propria senza sovrapposizioni.
Per quanto riguarda invece gli effetti sonori, per evitare problemi con la musica e per il funzionamento lievemente diverso che presentano, inseriamo una classe Sfx che genera quindi un suono che si sovrappone alla musica ma che può essere gestito indipendentemente.

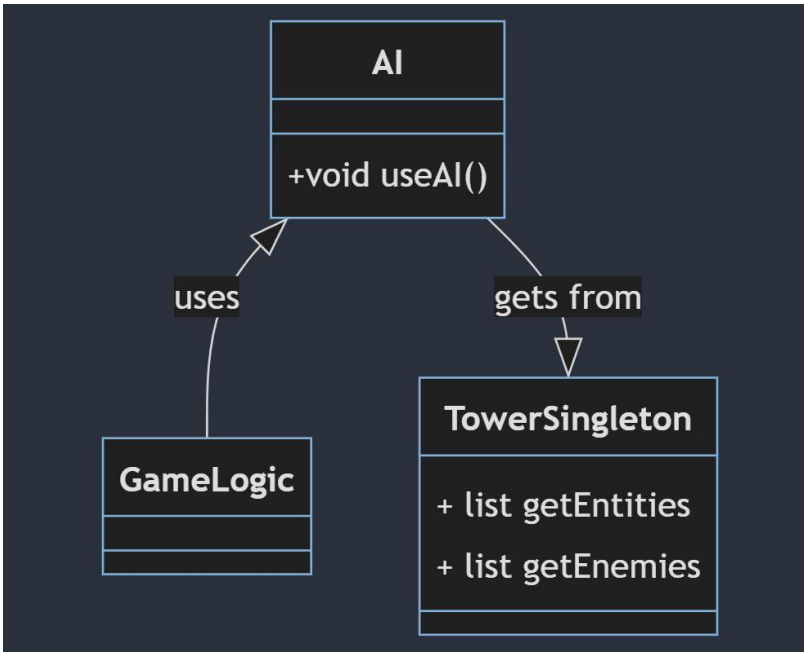
MINOCCARI

automazione del comportamento delle entità

Problema: Il gioco richiede da parte dell'utente solo un conivolgimento strategico e non diretto, perciò mi si è presentata la necessità di rendere le entità autonome cioè in grado di compredere quale decisione prendere in base all'ambiente circostante.

Soluzione: Per soddisfare questa necessità abbiamo creato una classe AI che mantiene come un paramentro una istanza di TowerSingletone da cui

vengono prese le liste di entità nemiche ed alleate presenti sul campo di battaglia. Da queste liste si analizza una per una dove si trova l'entità in questione e cosa la circonda in modo da scegliere uno dei comportamenti prestabiliti cioè attaccare, avanzare o fermarsi.
Un esempio e' il caso di essere il primo in fila delle proprie truppe, l'entità dovrà avanzare finchè il suo hitbox o rangebox non verrà triggerato o non si troverà troppo distante dalla base.



CAPITOLO 3

SVILUPPO

3.1 TESTING AUTOMATIZZATO

Nel contesto del progetto towerdefense abbiamo scelto di sottoporre a test automatizzato tutte le entità appartenti alle classi MovingEntity, RangedEntity e Projectile attraverso la suite di test specifica JUnit 5.
Abbiamo testato ogni metodo di ognuna delle superclassi elecante sopra per ogni caso specifico di implementazione (MovingEntity: Barbarian, Knight, Goblin... ; RangedEntity: Archer, Turret ecc) attraverso l'utilizzo del metodo assertEquals() per confrontare il risultato dei test specifici con i valori attesi.

3.2 METODOLOGIA DI LAVORO

Nelle prime fasi di lavoro abbiamo delineato uno scheletro del progetto da seguire e ci siamo divisi le parti (poi dovute ricalibrare dopo l'abbandono di un membro del gruppo).
Il DVCS è stato usato in modo molto lineare, con merge molto frequenti.

BADIOLI

Dal punto di vista della logica del programma mi sono occupato principalmente di realizzare le classi GameLoop, WaveManagerSingleton, FinalMove, e le funzioni di salvataggio e lettura dei punteggi su file.
Dal punto di vista grafico invece ho realizzato i pannelli di regole, menù e finale (esclusa la musica e negli ultimi due le immagini di sfondo) e implementato gli sprite del Wizard.
Altre cose di cui mi sono occupato è stato il code refactoring generale del programma e la documentazione javadoc.

PETRASSI

Il ruolo a me assegnato riguardava inizialmente la gestione dello spawn e despawn delle entità e l'implementazione della musica di sottofondo.
Nello svolgere nel dettaglio il lavoro, oltre ai punti sopra riportati, mi sono poi occupato personalmente della gestione delle diverse entità e delle relative immagini ad esse assegnate, ricoprendo dunque la parte grafica del progetto.

Ho contribuito nella gestione del "game panel", per quanto concerne i pulsanti assegnati alle entità stesse e il loro comportamento.

MINOCCARI

Per quanto riguarda la parte logica del progetto mi sono occupato di scrivere le classi Entity, MovingEntity, RangedEntity, Projectile e la classe AI per gestirne l'automazione.
Per quanto riguarda la parte grafica mi sono occupato degli sprite dell'Archer, del Goblin e della Turret .
Per finire mi sono occupato di gradleizare il progetto.

3.3 NOTE DI SVILUPPO

PETRASSI

Utilizzo della libreria javax.imageio.ImageIO e java.awt.image.BufferedImage
Attraverso l'implementazione della seguente funzione " void updateSprite(String activity)", collocata nella classe "MovingEntity" mi è stato possibile restituire il giusto percorso dove reperire lo sprite da riutilizzare in quel momento sulla data entità.

https://github.com/FilippoBadioli/OOP22-towerdefense/blob/4a967e1b4d0411394034a6abb119a4d6deef0e38/app/src/main/java/towerDefense/entities/api/MovingEntity.java#LL73C5-L108C6

Utilizzo della libreria javax.sound.sampled.*
Utilizzata per l'implementazione della musica e degli effetti sonori.

https://github.com/FilippoBadioli/OOP22-towerdefense/blob/eedded9723afc711b267b596e03116ef3d2d3a7f/app/src/main/java/towerDefense/game/impl/Music.java#L3

BADIOLI

Utilizzo delle librerie java.nio.file e java.awt.Desktop

La lettura/scrittura su file è stata realizzata grazie alle sotto classi Paths, Path, Files e Desktop e ai loro relativi metodi per poter accedere alla posizione del file all'interno del filesystem e poterlo aprire e/o scrivere

https://github.com/FilippoBadioli/OOP22-towerdefense/blob/4a967e1b4d0411394034a6abb119a4d6deef0e38/app/src/main/java/towerDefense/game/impl/EndPanel.java#LL46C1-L63C1

https://github.com/FilippoBadioli/OOP22-towerdefense/blob/4a967e1b4d0411394034a6abb119a4d6deef0e38/app/src/main/java/towerDefense/game/impl/MenuPanel.java#LL42C1-L50C12

Utilizzo di lambda functions per tutti i pulsanti presenti nei Panels, qui di seguito ne riporto alcuni esempi:

https://github.com/FilippoBadioli/OOP22-towerdefense/blob/4a967e1b4d0411394034a6abb119a4d6deef0e38/app/src/main/java/towerDefense/game/impl/EndPanel.java#LL51C9-L62C12

https://github.com/FilippoBadioli/OOP22-towerdefense/blob/4a967e1b4d0411394034a6abb119a4d6deef0e38/app/src/main/java/towerDefense/game/impl/GamePanel.java#LL64C8-L98C1

https://github.com/FilippoBadioli/OOP22-towerdefense/blob/4a967e1b4d0411394034a6abb119a4d6deef0e38/app/src/main/java/towerDefense/game/impl/GamePanel.java#LL113C9-L117C

https://github.com/FilippoBadioli/OOP22-towerdefense/blob/4a967e1b4d0411394034a6abb119a4d6deef0e38/app/src/main/java/towerDefense/game/impl/RulePanel.java#LL28C8-L33C12

MINOCCARI

Utilizzo di generics per metodi che accomunino diversi tipi di entità.

https://github.com/Giominoccari/OOP22-towerdefense/blob/eedded9723afc711b267b596e03116ef3d2d3a7f/app/src/main/java/towerDefense/gameLogic/impl/GameLogicImpl.java#L23-L36

CAPITOLO 4

COMMENTI FINALI

4.1 AUTOVALUTAZIONE

PETRASSI

Nel fornire un'autovalutazione quanto più oggettiva possibile, ciò che ritengo necessario far emergere in primis, è l'insegnamento acquisito mediante lo svolgimento del lavoro in gruppo.

Da questo lavoro di gruppo traggo come risultato fondamentale l'aver compreso le difficoltà che sorgono nell'ambito della collaborazione, per quanto riguarda il riuscire a mettere insieme esigenze e modus operandi completamente distinti. Come in un puzzle, è stato necessario trovare ogni singolo pezzo che riuscisse a incastrarsi in una cornice coerente e continua, con l'eccezione che a mettere mano su ogni pezzo non vi sia stata un'unica persona. La costanza, il dialogo e l'ascolto hanno fatto in modo che seppur ogni singolo componente del gruppo abbia avuto un proprio principale compito nella progettazione, tutti gli altri vi abbiano in qualche modo messo mano e dunque preso parte, facendo si che il risultato fosse il più omogeneo auspicabile.
Nonostante sia stata per me una delle primissime esperienze in un progetto di tale portata e complessità, ritengo di essermi impegnato nel mettere insieme le conoscenze acquisite lungo la durata dell'intero corso, ed insieme al background fornito dalle altre materie e alla mia curiosità di andare oltre l'insegnamento frontale, complessivamente mi ritengo soddisfatto del lavoro.

Essendo però questa una valutazione oggettiva, è necessario che faccia luce anche sui punti oscuri. Non posso ritenere che si tratti di una vera e propria dimostrazione di esser un capace progettista, ma è sicuramente un punto di partenza, dal quale solo l'esperienza e la continua dedizione riusciranno a mettere insieme quello che ho scoperto essere un mio forte interesse: la progettazione di videogiochi.

BADIOLI

Non avendo mai sviluppato un progetto di gruppo (sopratutto di queste dimensioni) e avendo quasi zero esperienze pregresse di programmazione prima del corso sono sicuro che molti degli strumenti da utilizzare per lo sviluppo di questo progetto io non sia riuscito a sfruttarli a dovere.
L'inesperienza ha impattato molto sopratutto nella fase di design iniziale, non avendo idea di quali problematiche potessero nascere durante la realizzazione del software, anche se poi durante la realizzazione pratica del progetto sono riuscito a realizzare tutto ciò che dovevo fare senza grosse complicazioni (anche se sicuramente molte cose sono state risolte in modi subottimali).
Faccio però tesoro dell'esperienza maturata durante lo sviluppo di questo progetto, che mi sarà sicuramente di grande aiuto in futuro.

MINOCCARI

Durante il progetto informatico relativo a TowerDefense, ho avuto l'opportunità di lavorare come parte di un team collaborativo. Il mio ruolo principale era concentrarmi sulle entità di gioco e sulla loro automazione. Durante il processo, ho avuto modo di sperimentare diverse sfide e imparare nuovi concetti.

In quanto membro del team, ho contribuito attivamente alle discussioni sul design generale del gioco e ho condiviso le mie idee sull'implementazione delle entità. Abbiamo lavorato insieme per definire gli obiettivi e le specifiche delle entità, tenendo conto delle esigenze dei giocatori e dell'esperienza di gioco desiderata.

L’esperienza più formate è stato lavorare in gruppo ed essendo una formazione necessaria per lavorare in questo campo credo che mi abbia fatto crescere molto

4.2 DIFFICOLTA' INCONTRATE E COMMENTI AI DOCENTI

BADIOLI

Come già fatto presente al docente, durante il lavoro, il gruppo ha riscontrato la problematica di un membro che, inizialmente, non faceva la sua parte e dopo numerose interpellazioni si è fatto da parte dal progetto.
Questo ha causato non pochi disagi, primo tra tutti a livello temporale: il progetto è slittato di deadline, arrivando così molto vicino alla sessione di esami estiva con conseguente aumento di impegni universitari da parte dei membri del gruppo.
Ha causato anche disagi tecnici, in quanto in 3 persone dovevamo ora occuparci di un progetto pensato per 4, cercando di renderlo comunque più completo possibile rispetto alla proposta originaria, e psicologici, dato che l'abbandono di un membro del gruppo dopo diverse settimane di mancate risposte ha notevolmente calato l'enfasi generale verso questo progetto e portato anche a maggiori stress e pressioni.
Mi dispiace puntualizzuare un problema senza essere in grado di suggerirne una possibile soluzione, ma spero che si possa trovare un modo per continuare a svolgere questa attività a gruppi abbattendo questa problematica (dato che rischia di impattare gravemente su carriere universitarie di altri studenti).

Spero infine di ricevere più feedback possibile su ciò che è stato "mal realizzato" o "mal progettato" in modo da sapere su quali aspetti della programmazione ad oggetti ho maggiori lacune.

GUIDA UTENTE

All'avvio del gioco sarà possibile scegliere tra 3 menù: "Start Game", "Best Scores" e "Rules"
Cliccando su "Start Same" il gioco comincerà, cliccando su "Best Scores" si aprirà un file contenente i punteggi migliori salvati mentre cliccando su "Rules" si aprirà una finestra con il regolamento del gioco (da cui è possibile tornare indietro con il pulsante "Back").

Una volta cominciato il gioco inizieranno ad arrivare ondate di nemici, per poterli sconfiggere bisognerà evocare alleati e/o potenziamenti tramite i pulsanti trovati in cima alla finestra di gioco, al costo di monete di gioco.
Attualmente le unità/potenziamenti presenti sono:

- -Barbaro: unità lenta, con molti punti salute e un discreto attacco
- -Cavaliere: unità veloce, con pochi punti salute ma molti danni
- -Arcere: unità veloce, con pochissimi punti salute ma molti danni e una gittata media
- -Torretta: cannone posto sulla cima della torre, è un'unità monouso con gittata molto ampia (se dei nemici arrivano alla torre colpiranno prima la torretta fino a distruggerla, poi la torre)
- -Potenziamento: chiamato "Double Allies" permetterà di evocare il doppio degli alleati al costo di uno solo per un certo lasso di tempo

Una volta evocata un'unità bisognerà aspettare qualche secondo prima di poterne evocare un'altra
Il potenziamento ha invece un tempo di ricarica molto più lungo
La partita finirà quando i punti salute della torre scendono a zero oppure cliccando il pulsante "Surrender", a quel punto si verrà reindirizzati al menù di fine partita.
Da qui si potrà salvare il proprio punteggio inserendo il proprio nome e poi cliccando "Save Score" e/o si potrà chiudere il gioco