**Documentation – Space Invaders**

**What went well?**

Development as a whole was generally smooth. While the fundamentals (collision, placement and movement) took time to implement, everything following that came quite quickly and I was able to make progress at a reasonable rate. There also weren’t any bugs that took major chunks out of development time and as a result, things began to take shape quickly and development never came to a complete standstill.

Expanding on the state machine also went surprisingly well, as it was just a matter of creating different states and setting the game state to them in response to given stimuli. This was all stuff that was either in the starting code or that I already knew.

**What made things difficult?**

The hardest part of development was getting the initial ball rolling and getting the foundations laid out. While development never hit a complete standstill, it did slow down while I got to better grips with the system and hot to use the renderer to spawn the different sprites and program them to move with input. Enemies also took some time to correctly implement, but ultimately it was just a case of figuring out the correct way to move them all as a group.

**What Could be improved?**

While I am satisfied with the end product, there are certainly ways I could’ve expanded on the implementation and design of the game to improve it. Even before features, I could have followed the coding style from the beginning and as I got deeper in without acknowledging it, I found out it would only get more time consuming to implement.

I could also have made some functionality more data driven. A prime example of this is in my collision and arrays, instead of having a variable to store the numbers needed for different mathematics or loops, I hard coded it in order to just get it working as quickly as possible. While this did work in the base sense of completing the assignment, it is poor practice and something I should take care to implement early on in future.

For additional features, I could have added an extra enemy type and an audio component to make the game a bit more engaging. There was also room to add power ups to branch the game out from a Space Invaders clone to something more.

In terms of expanding on current features, I could have made the barriers decay to make the game more like the real Space Invaders and ensure that the game has progression and that the player is in more danger as the game goes on. I could also have added calculations for the enemies to simulate them strategically firing based on the location of the player.

I feel all these critiques mount to one big problem I had during development: Time management and planning. I jumped straight into implementation in a panic to begin making progress and meet deadlines and as a result I ended up preparing my work poorly and having to hack in a lot of solutions to my problems as I went along, leading to code that while functional, could have been a lot better and allowed time for extra features to be added.

**Conclusion**Ultimately, I think I did a satisfactory job in fulfilling the criteria set out in the assignment brief. However, as mentioned there was a lot of room for creativity and ambition that I missed out on due to poor time management at the beginning of development. I feel I’ve learned a lot from this development cycle and in future I will aim to dedicate enough time to planning to allow for much more efficient code.

**UML Diagram on Next Page**

|  |
| --- |
| **Class Barrier**  -barrierSprite: unique\_ptr<ASGE::Sprite>  -renderer: shared\_ptr<ASGE::  Renderer> |
| +Barrier(temp\_renderer)  +~Barrier()  +render(): bool |

|  |
| --- |
| **Class Game**  -framesPassed: int  - callback\_id: int  -exit: bool  -gameStart: bool  -baseX: int  -randomNumber: int  -baseY: int  -barrierBaseX: int  -enemyX: int  -enemyY: int  -level: int  -nextLevel: bool  -menuSprite: unique\_ptr<Sprite>  -menuSprite2: unique\_ptr<Sprite>  -enemies: vector<unique\_ptr<Enemy>>  -barriers: vector<unique\_ptr<Barrier>>  -player\_one: unique\_ptr<Player> |
| +InvadersGame()  +~InvadersGame()  -processGameActions(): void  -input(int key, int action) const: void  +run() override: virtual bool  +shouldExit() const: bool  +render(): void  +init(): virtual bool  +drawFrame(): virtual void  +createEnemyArray(): void  +createBarrierArray(): void  +CheckForCollision(float, float, float, float): bool  +CheckForBarrierCollision(float, float, float, float): bool  +RespawnEnemies(bool newRound): void |

|  |
| --- |
| **Class Player**  -lives: int  -score: int  -moveSpeed: float  -playerSprite: unique\_ptr<ASGE::Sprite>  -renderer: shared\_ptr<ASGE::  Renderer>  +bullets: vector<unique\_ptr<Bullet>> |
| +Player(temp\_renderer)  +~Player()  +Attack() Override: void  +MoveLeft(): void  +MoveRight(): void  +render(): bool  +createBulletArray(): void  +GetScore(): float  +SetScore(float newScore): void  +GetLives(): int  +SetLives(int newLives): void |

|  |
| --- |
| **Class Enemy**  -movingRight: bool  -moveSpeed: float  -startPosX: int  -startPosY: int  -renderer: shared\_ptr<ASGE::  Renderer>  -enemySprite: unique\_ptr<ASGE::  Sprite>  +enemyBullets: vector<<unique\_ptr<Bullet>> |
| +Enemy(temp\_renderer)  +~Enemy()  +Attack() Override: void  +Move(EnemyAction enemy\_action): void  +render(): void  +GetStartPosX(): int  +GetStartPosY(): int  +GetMoveSpeed(): float  +SetStartPosX(int newStartPosX): void  +SetStartPOsY(float newStartPosY): void  +SetMoveSpeed(float newMoveSpeed): void  +createBulletArray(): void |

|  |
| --- |
| **Class GameObject**  -isAlive: bool  -posX: float  -posY: float  -height: float  -width: float |
| +GameObject()  +~GameObject()  +Attack(): void  +SetPosX(float newPosX): void  +SetPosY(float newPosY): void  +SetWidth(float newWidth): void  +SetHeight(float newHeight): void  +SetIsAlive(bool isItAlive): void  +GetPosX(): float  +GetPosY(): float  +GetWidth(): float  +GetHeight(): float  +GetIsAlive(): bool |

|  |
| --- |
| **Class Bullet**  -renderer: shared\_ptr<ASGE::  Renderer>  -bulletSprite: unique\_ptr<ASGE::  Sprite> |
| +Bullet(temp\_renderer)  +~Bullet()  +Move(): void  +EnemyMove(): void  +render(): bool |