BMS COLLEGE OF ENGINEERING BENGALURU

Autonomous Institute, Affiliated to VTU

SPC AAT Report on

TOWER DEFENCE GAME

Submitted in partial fulfillment of the requirements for AAT

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by:

SUSHANT KUMAR SINGH

TUSHAR KHATRI

Department of Computer Science and Engineering

BMS College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
2025-2026

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, SUSHANT KUMAR SINGH AND TUSHAR KHATRI students of 1st Semester, B.E, Department of AIML, BMS College of Engineering, Bengaluru, hereby declare that, this AAT Project entitled " **TOWER DEFENCE GAME** " has been carried out in Department of CSE, BMS College of Engineering, Bengaluru during the academic semester Oct 2025 – Jan 2026. We also declare that to the best of our knowledge and belief, the AAT Project report is not from part of any other report by any other students.

Student Name	**Student Signature**
1. SUSHANT KUMAR SINGH	
2. TUSHAR KHATRI	

BMS COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the AAT Project titled "**TOWER DEFENCE GAME"** has been carried out by SUSHANT KUAMR SINGH (1BM25AI245) and TUSHAR KHATRI(1BM25AI164) during the academic year 2025-2026.

Signature of the Faculty in Charge

Table of Contents

1. INTRODUCTION

The core elements of a space shooter game in C include:

Game Loop: The game uses a simple while-loop game loop where the program repeatedly processes input, updates the game state, and renders output each wave until the base health reaches zero and the game enters the GAME_OVER state

Player Control: The player control used to run the game is

- Pressing Enter to start the game from the menu
- Typing a number (0, 1, or 2) and pressing Enter to choose whether to build a tower or skip during each wave

So, the player controls the game by entering numeric choices through the keyboard using scanf and getchar

Objects and Variables: The game uses Tower and Enemy objects (structures) to represent defence and attackers, and variables such as state, wave, money, baseHealth, base Enemy Health, and tower Count to control the game flow, progression, and player resources.

Rendering: is done by printing the current game information to the console, specifically displaying the wave number, player money, and base health using printf

Console I/O: Simple text-based representation within the console window**.**

Collision Detection: This game does not implement collision detection, because enemies and towers are not positioned in a coordinate system; instead, all tower damage is applied directly to all enemies each wave without checking positions

Game State Management: The game manages its state using the Game State num (MENU, PLAYING, GAME_OVER) and a variable state that controls which part of the game loop runs, allowing the program to switch between showing the menu, playing the game, and ending the game when the base health reaches zero.

2. ALGORITHM

Step 1: Start the Game

Set state = MENU

Set wave = 1

Set money = 100

Set baseHealth = 5

Set baseEnemyHealth = 30

Set towerCount = 0

Create arrays:

Towers[MAX_TOWERS]

enemies[MAX_ENEMIES]

Step 2: Display Menu

Display game title and instructions

Wait for user to press Enter

Change game state to PLAYING

Step 3: Game Loop (Repeat until state = GAME_OVER)

Step 4: Render Game Status

Step 5: Display-

Current wave number

Available money

Base health

Step 6: Tower Placement

If towerCount < MAX_TOWERS:

Display tower options:Set state = GAME_OVER

Else:

Increase wave number by 1

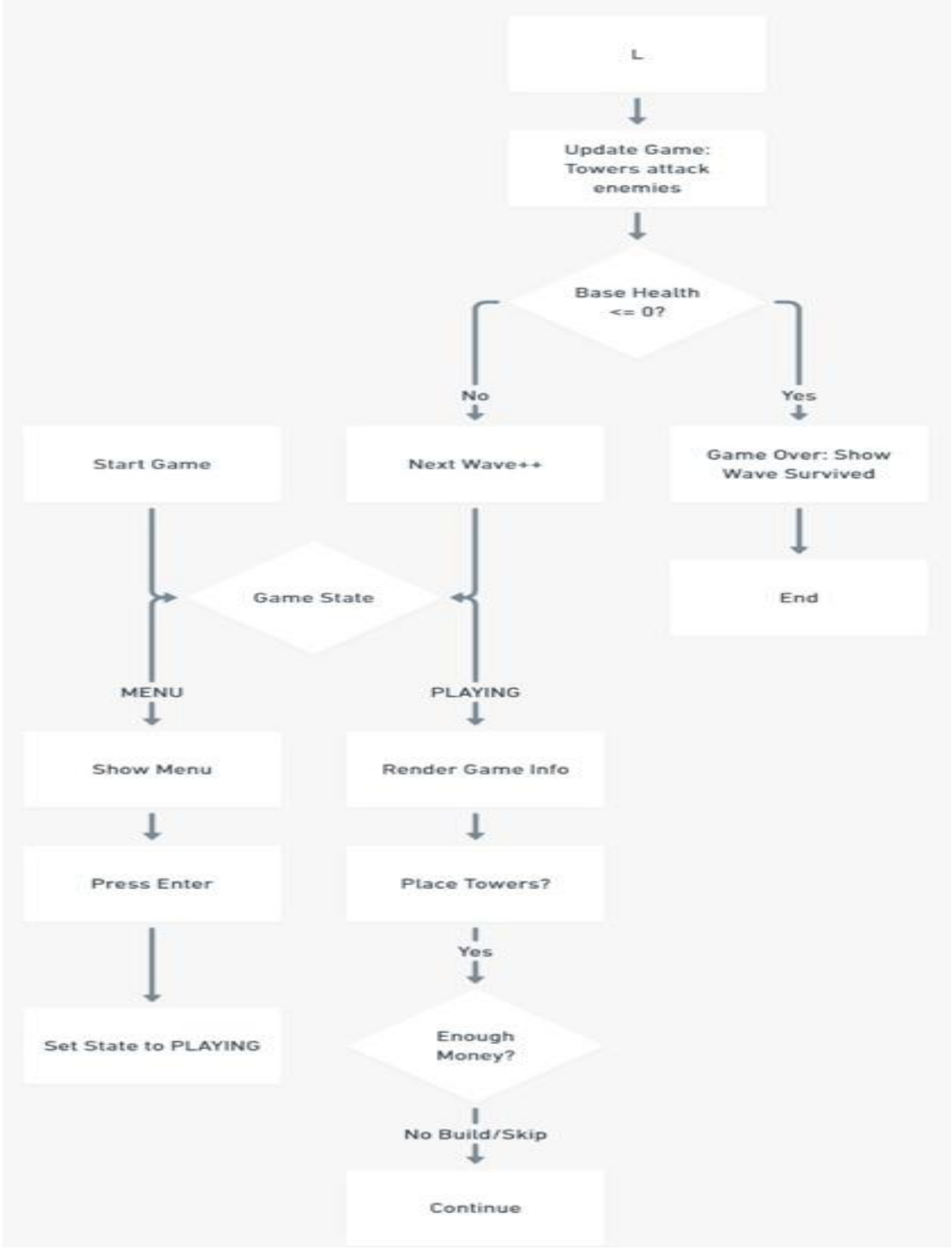
Step 7: End Game

Display "GAME OVER"

Display final wave reached

Step 8: Stop

3. FLOWCHART


L
Update Game: Towers attack enemies
Base Health <= 0?
No
Yes
Start Game
Next Wave++
Game Over: Show Wave Survived
Game State
End
MENU
PLAYING
Show Menu
Render Game Info
Press Enter
Place Towers?
Yes
Set State to PLAYING
Enough Money?
No Build/Skip
Continue

4. SOURCE CODE

```
#include <stdio.h>

#include <stdlib.h>

#define MAX_TOWERS 5

#define MAX_ENEMIES 5

// -------- GAME STATES --------

typedef enum {

MENU,

PLAYING,

GAME_OVER

} GameState;

// -------- TOWER TYPES --------

typedef enum {

ARCHER = 1,

CANNON

} TowerType;
```

```
// -------- STRUCTURES --------

typedef struct {

TowerType type;

int damage;

int level;

} Tower;

typedef struct {

int health;

int alive;

} Enemy;

// -------- FUNCTION PROTOTYPES -

void showMenu();

void placeTowers(Tower towers[], int
*towerCount, int *money);

void spawnEnemies(Enemy enemies[],
int wave, int baseHealth);

void updateGame(Tower towers[], int
towerCount, Enemy enemies[],
```

```
int enemyCount, int *money, int
*baseHealth);

void renderGame(int wave, int money,
int baseHealth);

int totalTowerDamage(Tower towers[],
int towerCount);

int main() {

GameState state = MENU;

Tower towers[MAX_TOWERS];

Enemy enemies[MAX_ENEMIES];

int towerCount = 0;

int wave = 1;

int money = 100;

int baseHealth = 5;

int baseEnemyHealth = 30;



while (state != GAME_OVER) {

// -------- INPUT --------

if (state == MENU) {
```

```
showMenu();

state = PLAYING;

}

if (state == PLAYING) {

renderGame(wave, money, baseHealth);

placeTowers(towers, &towerCount,
&money);

spawnEnemies(enemies, wave,
baseEnemyHealth);

updateGame(towers, towerCount,
enemies,

wave + 1, &money, &baseHealth);

if (baseHealth <= 0) {

state = GAME_OVER;

} else {

wave++;

}

}

}
```

```
printf("\n💀 GAME OVER! You
survived until Wave %d\n", wave);

return 0;

}

// -------- FUNCTIONS --------

void showMenu() {

printf("===== TOWER DEFENCE
GAME =====\n");

printf("Defend your base from incoming
enemies!\n");

printf("Press Enter to start...");

getchar();

}

void placeTowers(Tower towers[], int
*towerCount, int *money) {

int choice;

if (*towerCount >= MAX_TOWERS)
return;

printf("\nMoney: %d\n", *money);
```

```
printf("1. Build Archer Tower (Cost:
50)\n");

printf("2. Build Cannon Tower (Cost:
70)\n");

printf("0. Skip\n");

printf("Choice: ");

scanf("%d", &choice);

if (choice == 1 && *money >= 50) {

towers[*towerCount] =
(Tower){ARCHER, 10, 1};

(*towerCount)++;

*money -= 50;

}

else if (choice == 2 && *money >= 70) {

towers[*towerCount] =
(Tower){CANNON, 18, 1};

(*towerCount)++;

*money -= 70;

}
```

```
}

void spawnEnemies(Enemy enemies[],
int wave, int baseHealth) {

for (int i = 0; i < wave + 1; i++) {

enemies[i].health = baseHealth + (wave *
10);

enemies[i].alive = 1;

}

}

void updateGame(Tower towers[], int
towerCount, Enemy enemies[],

int enemyCount, int *money, int
*baseHealth) {

int damage = totalTowerDamage(towers,
towerCount);

for (int i = 0; i < enemyCount; i++) {

if (!enemies[i].alive) continue;

enemies[i].health -= damage;

if (enemies[i].health <= 0) {

enemies[i].alive = 0;
```

```
    *money += 20;
    printf("Enemy defeated! +20 money\n");
    } else {
    (*baseHealth)--;
    printf("Enemy reached base! Base health:
    %d\n", *baseHealth);
    }
    }
    }
 Void renderGame(int wave, int
money, int baseHealth) {
    printf("\n---------------------------\n");
    printf("Wave: %d | Money: %d | Base
    Health: %d\n",
    wave, money, baseHealth);
    printf("-------------------------\n");
    }
    int totalTowerDamage(Tower towers[],
    int towerCount) {
```

```
    int total = 0;
    for (int i = 0; i < towerCount; i++) {
    total += towers[i].damage;
    }
    return total;
}
```

5. RESULTS

```
Output
===== TOWER DEFENCE GAME =====
Defend your base from incoming enemies!
Press Enter to start...

------------------------------
Wave: 1 | Money: 100 | Base Health: 5
------------------------------

Money: 100
1. Build Archer Tower (Cost: 50)
2. Build Cannon Tower (Cost: 70)
0. Skip
Choice: 1
Enemy reached base! Base health: 4
Enemy reached base! Base health: 3
```

```
------------------------------
Wave: 2 | Money: 50 | Base Health: 3
------------------------------

Money: 50
1. Build Archer Tower (Cost: 50)
2. Build Cannon Tower (Cost: 70)
0. Skip
Choice: 2
Enemy reached base! Base health: 2
Enemy reached base! Base health: 1
Enemy reached base! Base health: 0

💀 GAME OVER! You survived until Wave 2


=== Code Execution Successful ===
```

REFERENCES

1. **Kernighan, B. W., & Ritchie, D. M.**
   The C Programming Language, 2nd Edition, Prentice Hall, 1988.
2. **Balagurusamy, E.**
   Programming in ANSI C, McGraw Hill Education, India.
3. **Kanetkar, Y.**
   Let Us C, BPB Publications.
4. **Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C.**
   Introduction to Algorithms, MIT Press.
5. **ISO 5807:1985**
   Information Processing — Documentation Symbols and Conventions for Data, Program and System Flowcharts.
6. **B. Ram**
   Computer Fundamentals, New Age International Publishers.
7. **C Standard Library Documentation**
   <stdio.h>, <string.h>, <time.h> — for input/output, string handling, and time measurement.
8. **Using chat gpt and google**