# BEERMAN

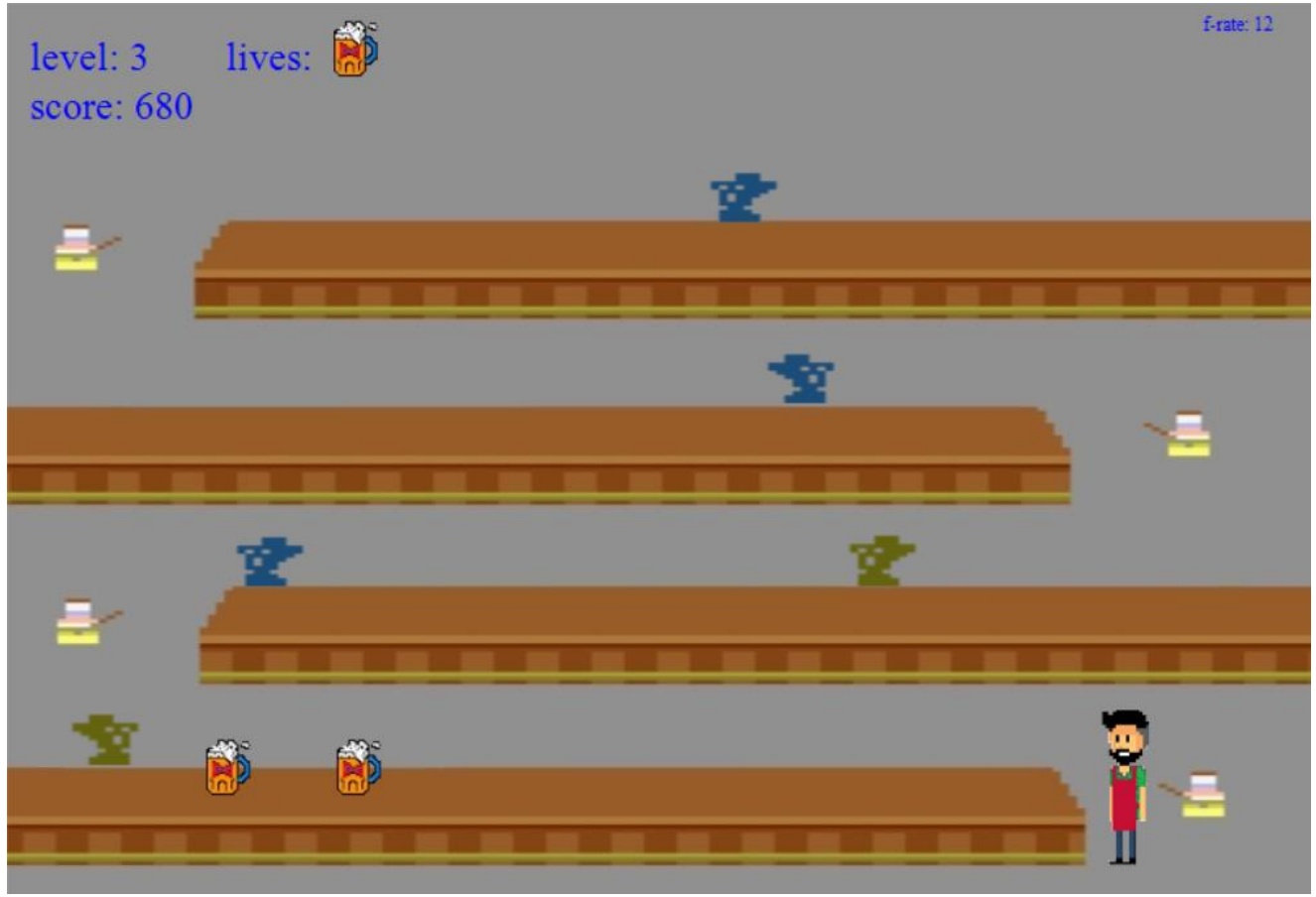
di Locatelli Celeste

# Trama del gioco

Beerman è ambientato in un bar o locale con 4 tavoli di lunghezza variabile. Il protagonista del gioco è il barista e lo scopo è servire tutti i clienti che si presentano sullo schermo. Ogni cliente servito aumenterà il punteggio di 10 unità, i migliori 20 punteggi verranno visualizzati al termine del gioco.

# Regolamento:

La partita termina quando il giocatore ha perso tutte le vite, queste diminuiscono nel momento in cui una birra arriva al limite di un tavolo oppure quando un cliente si avvicina troppo al nostro personaggio. Alcuni sono impazienti mentre altri possono richiedere più birre, questi fattori sono casuali e rendono il gioco più veloce ma anche più divertente.



# Istruzioni:

## Fase di avvio:

- Premere spazio per avviare il gioco

## Fase di gioco:

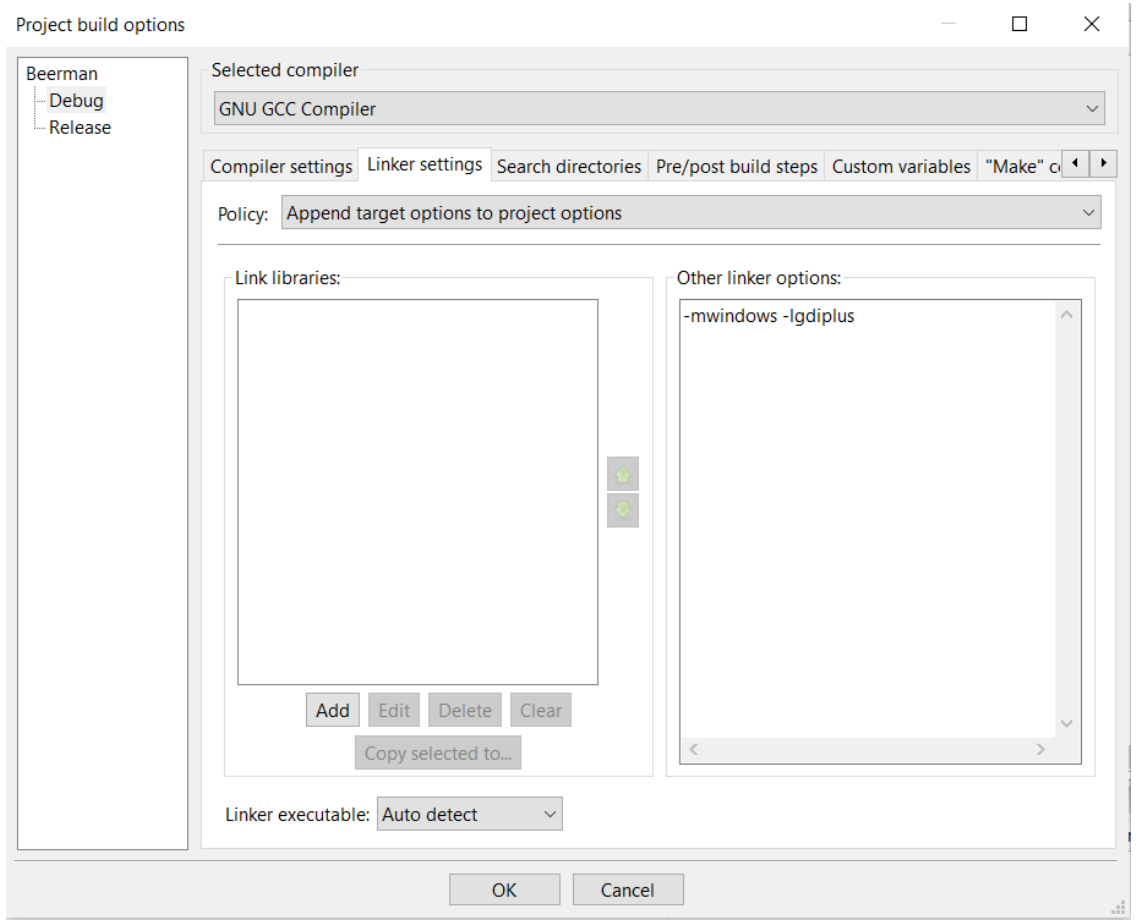
- Usa le freccette per spostarti tra i tavoli
- Premi lo spazio per lanciare una birra
- Premi lo spazio per passare da un livello all'altro

## Fase conclusiva:

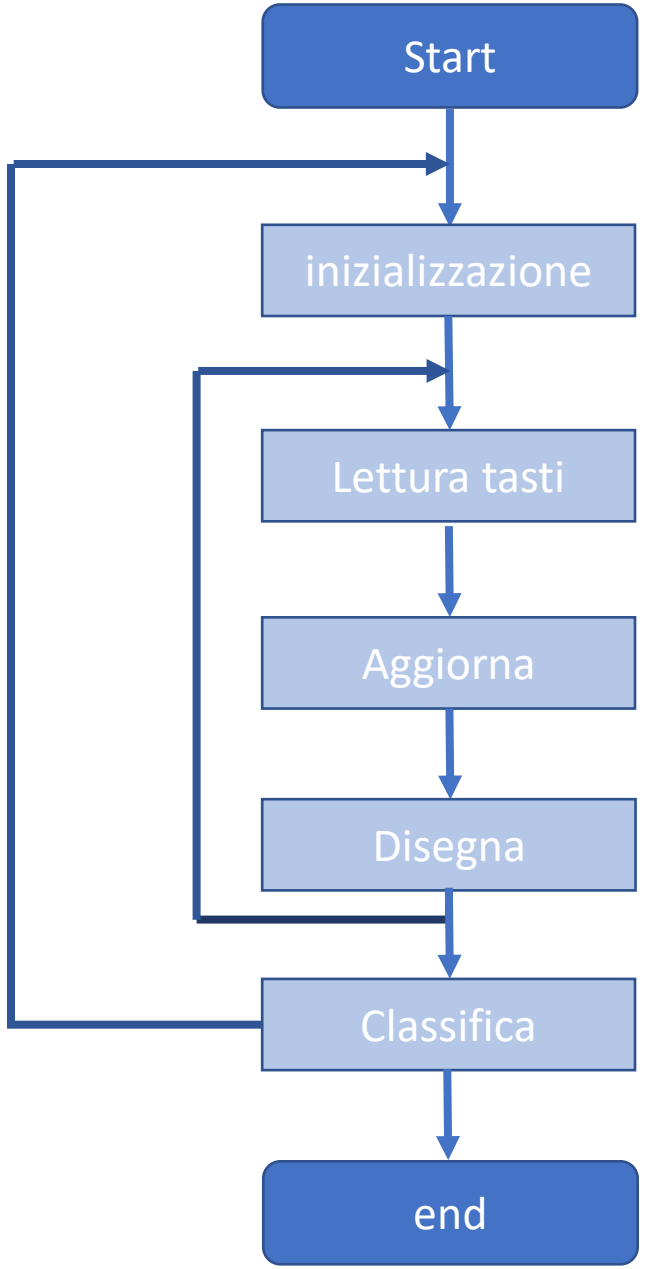
- Classificato:
  - Digita il tuo nickname e premi invio
- Non classificato:
  - Premi lo spazio per rigiocare oppure chiudi la finestra per uscire dal gioco

# Ambiente di sviluppo

- Il gioco è stato programmato in C++ utilizzando l'ambiente di sviluppo CodeBlocks.
- L'applicazione è a finestra windows e utilizza grafica 2D supportata dalla libreria grafica GDIPlus
- Per connettere la libreria GDIPlus ho dovuto impostare la seguente configurazione in CodeBlocks.



# Diagramma di flusso



# Lettura dei tasti

Per raggiungere questo scopo mi sono servito della funzione WindowsProcessMessages che viene invocata all’attivazione di un evento come ad esempio la pressione su un tasto. Il comportamento di ogni singolo tasto è stato ottenuto attraverso uno switch della variabile param dal quale è stato possibile definire gli effetti dei tasti sul gioco nei vari momenti.

# Risoluzione del flickering

Il flickering è un problema che emerge spesso nella programmazione dei giochi. Si manifesta quando vengono stampati a schermo oggetti in movimento che dovrebbero spostarsi con fluidità ma a causa del  flickering si muovono a scatti. Per risolvere questo inconveniente ho utilizzato la tecnica del doublebuffering che detto semplicemente, compone l'immagine da visualizzare in memoria e solo alla fine viene riportata sullo schermo, evitando l'effetto di ridisegno.

```
void initializeGraphics(){
    // Double buffering, borrowed from GDI-provided wrapper in next assignment
    RECT windowSize = { 0,  0, WIDTH, HEIGHT};
    HDC hdc = GetDC(window);
    backbufferDC = CreateCompatibleDC(hdc);

    backbufferBitmap = CreateCompatibleBitmap(hdc, WIDTH, HEIGHT);
    // Store old object so that we don't leak.
    oldObject = SelectObject(backbufferDC, backbufferBitmap);

    SetBkMode(backbufferDC, TRANSPARENT);
}

void switchBuffer(){
    // Blit-block transfer to the main device context
    HDC windowDC = GetDC(window);
    BitBlt(windowDC, 0, 0, WIDTH, HEIGHT, backbufferDC, 0, 0, SRCCOPY);
    ReleaseDC(window, windowDC);
}
```

# Classifica

Per la classifica ho scelto di visualizzare i 20 punteggi migliori. Se il risultato di una partita rientra tra questi, si richiede l'inserimento di un nome che sarà composto da sole lettere. Prima di inserire il nick leggo la classifica precedente e la aggiorno se ci sono entrato, inizialmente il nick è costituito da una serie di underscore che verrà successivamente sostituito dal nick scelto dall'utente, per fare ciò ho utilizzato nuovamente la funzione WindowsProcessMessages. Terminata la scrittura del nick è necessario premere l'invio per salvare il record ottenuto con il rispettivo nome.