

# Tekken DSL - Fighter Game Domain Specific Language

A C++ header-only DSL for creating turn-based fighting games inspired by Tekken.

---

## Build Instructions

### Requirements

- C++11 compatible compiler (g++, clang++, MSVC)
- UTF-8 source file support (for Greek character)

### Linux / macOS

```
# Simple build
g++ -std=c++11 -o game examples/main.cpp
g++ -std=c++11 -o example2 examples/example2.cpp

# Or with clang
clang++ -std=c++11 -o game examples/main.cpp

# Run
./game
./example2
```

### Windows (Command Prompt with MinGW/g++)

```
g++ -std=c++11 -o game.exe examples/main.cpp
g++ -std=c++11 -o example2.exe examples/example2.cpp

game.exe
example2.exe
```

### Windows (PowerShell with MinGW/g++)

```
g++ -std=c++11 -o game.exe examples/main.cpp
.\game.exe
```

### Windows (Visual Studio Developer Command Prompt)

```
cl /EHsc /std:c++14 /I include examples\main.cpp /Fe:game.exe
game.exe
```

## CMake (All Platforms)

```
mkdir build && cd build
cmake ..
cmake --build .

# Run (Linux/macOS)
./tekken_example

# Run (Windows)
.\Debug\tekken_example.exe
```

---

## How to Play

### Starting the Game

1. Run the compiled executable
2. The game displays available fighters
3. Player 1 selects a fighter by typing the exact name
4. Player 2 selects a fighter by typing the exact name
5. Both players can select the same fighter

### During Battle

Each round:

1. Player 1 sees their available abilities and selects one
2. After ability executes, both fighters' status is displayed
3. Player 2 sees their available abilities and selects one
4. After ability executes, both fighters' status is displayed
5. If a fighter's HP reaches 0, the other wins

### Important Notes

- Abilities are NOT global - each fighter can only use abilities they learned via DEAR...LEARN
- Out of ring fighters cannot act - if tagged out, a message is shown instead
- Out of ring fighters cannot receive damage - damage is blocked while out

### Sample Gameplay

```
-----FIGHTER THE GAME-----
Player1 select fighter:
-----
Lee
```

```

Jack-6
-----
Jack-6

Player2 select fighter:
-----
Lee
Jack-6
-----
Lee

~~~~~
Round 1
~~~~~

Jack-6(Player1) select ability:
-----
Head_Smash
Catch_A_Break
Bleeding_Bite
-----
Head_Smash

#####
Name: Lee
HP: 78
fighter enters the ring
#####

#####
Name: Jack-6
HP: 90
fighter enters the ring
#####

```

---

## DSL Syntax Guide

### Program Structure

```

#include "Tekken.h"

BEGIN_GAME

// Define abilities, fighters, and learn commands here

```

DUEL

END\_GAME

**Important:** No semicolons after CREATE statements!

---

### Creating Fighters

#### Single Fighter

```
CREATE FIGHTER {
    NAME: "FighterName",
    TYPE: "Rushdown",
    HP: 100
}
```

#### Multiple Fighters

```
CREATE FIGHTERS [
    FIGHTER {
        NAME: "Fighter1",
        TYPE: "Heavy",
        HP: 120
    },
    FIGHTER {
        NAME: "Fighter2",
        TYPE: "Evasive",
        HP: 90
    }
]
```

#### Fighter Types and Modifiers

Type	Outgoing Damage	Incoming Damage	Special
<b>Rushdown</b>	+15% to all, +20% vs Grappler	Normal	-
<b>Heavy</b>	Normal	-20% from all, -30% from Evasive	-
<b>Evasive</b>	+7% to all	-7% from all	-
<b>Grappler</b>	+7% on odd rounds	Normal	+5% max HP heal on even rounds

---

## Creating Abilities

### Single Ability

```
CREATE ABILITY {
    NAME: "AbilityName",
    ACTION: START
    // Commands here
    END
}
```

### Multiple Abilities

```
CREATE ABILITIES [
    ABILITY {
        NAME: "Ability1",
        ACTION: START
        DAMAGE DEFENDER 20
        END
    },
    ABILITY {
        NAME: "Ability2",
        ACTION: START
        HEAL ATTACKER 15
        END
    }
]
```

**Note:** Ability names cannot contain spaces.

---

## Teaching Abilities to Fighters

```
DEAR "FighterName" LEARN [
    ABILITY_NAME(Ability1)
    ABILITY_NAME(Ability2)
    ABILITY_NAME(Ability3)
]
```

**Important:** - Fighter name is in quotes - Ability names are NOT in quotes  
- NO commas between ABILITY\_NAME entries - Fighters can ONLY use abilities they have learned

---

## Action Commands

Command	Description
DAMAGE DEFENDER n	Deal n damage to opponent
DAMAGE ATTACKER n	Deal n damage to self
HEAL DEFENDER n	Heal opponent by n
HEAL ATTACKER n	Heal self by n
TAG DEFENDER ---	Remove opponent from ring
TAG DEFENDER _	Return opponent to ring
TAG ATTACKER ---	Remove self from ring
TAG ATTACKER _	Return self to ring
SHOW expression	Print to console

## Getter Functions

Function	Returns
GET_HP(DEFENDER/ATTACKER)	Current HP
GET_TYPE(DEFENDER/ATTACKER)	Fighter type string
GET_NAME(DEFENDER/ATTACKER)	Fighter name
IS_OUT_OF_RING(DEFENDER/ATTACKER)	Boolean

## Comparisons

```
GET_HP(DEFENDER) > 50
GET_TYPE(ATTACKER) == "Rushdown"
GET_HP(DEFENDER) <= 20
```

## Logical Operators

AND(condition1, condition2, ...)	// All must be true (2+ args)
OR(condition1, condition2, ...)	// At least one true (2+ args)
NOT(condition)	// Negation

## Control Flow

### IF/ELSE

```
IF GET_HP(DEFENDER) < 50 DO
    DAMAGE DEFENDER 35
ELSE_IF GET_HP(DEFENDER) < 80 DO
    DAMAGE DEFENDER 25
ELSE
    DAMAGE DEFENDER 15
END
```

**FOR Loop** (executes for N future rounds)

```
FOR 5 ROUNDS DO
    DAMAGE DEFENDER 8
END
```

**AFTER** (executes once after N rounds)

```
AFTER 2 ROUNDS DO
    TAG DEFENDER _
END
```

---

### Complete Example

```
#include "Tekken.h"

BEGIN_GAME

CREATE ABILITY {
    NAME: "Power_Strike",
    ACTION: START
    IF GET_HP(DEFENDER) > 50 DO
        DAMAGE DEFENDER 20
    ELSE
        DAMAGE DEFENDER 35
    END
}
CREATE ABILITY {
    NAME: "Healing_Wave",
    ACTION: START
    HEAL ATTACKER 25
}
CREATE FIGHTER {
    NAME: "Warrior",
    TYPE: "Rushdown",
    HP: 100
}
CREATE FIGHTER {
    NAME: "Tank",
    TYPE: "Heavy",
```

```

    HP: 120
}

DEAR "Warrior" LEARN [
    ABILITY_NAME(Power_Strike)
    ABILITY_NAME(Healing_Wave)
]

DEAR "Tank" LEARN [
    ABILITY_NAME(Power_Strike)
]

DUEL

END_GAME

```

---

## Project Structure

```

include/
    Tekken.h          # DSL implementation (header-only)
examples/
    main.cpp          # PDF example (Lee vs Jack-6)
    main_output.txt   # Expected output
    example2.cpp      # Custom example (Ryu vs Zangief)
    example2_output.txt # Expected output
CMakeLists.txt          # CMake build configuration
README.md               # This file
TECHNICAL.md            # Technical implementation details

```

---

## Troubleshooting

### Greek character not compiling

Ensure your source file is saved as UTF-8. In most editors: - VS Code: Bottom right → UTF-8 - Notepad++: Encoding → UTF-8

### Abilities not working

- Check that the fighter has learned the ability with DEAR...LEARN
- Abilities are NOT global - each fighter needs their own LEARN statement
- Verify ability name spelling matches exactly

### **Fighter not taking damage**

- Check if fighter is out of ring (TAG --- )
- Out of ring fighters cannot receive damage