Créer Votre Premier Projet

Guide pas-à-pas pour créer votre premier projet avec Jenga Build System.

Objectif

À la fin de ce tutoriel, vous aurez :

- ☑ Un projet C++ fonctionnel
- ☑ Compilation réussie
- ☑ Exécution de l'application
- ☑ Compréhension de base de Jenga

Temps estimé : 15 minutes

Prérequis

- Jenga installé
- Compilateur C++ (GCC, Clang, ou MSVC)
- Éditeur de texte ou IDE

Étape 1 : Créer la Structure du Projet

1.1 Créer le Répertoire

```
mkdir HelloJenga
cd HelloJenga
```

1.2 Structure Recommandée

```
HelloJenga/
├── hello.jenga          # Configuration du build
├── src/                 # Code source
│   └── main.cpp
├── include/             # Headers (optionnel)
└── README.md           # Documentation (optionnel)
```

Créons cette structure :

```
mkdir -p src include
```

Étape 2 : Écrire le Code

2.1 Créer main.cpp

Créer `src/main.cpp` :

```
#include <iostream>
#include <string>

int main(int argc, char** argv) {
    std::cout << "╔═══════════════════════════════════╗" << std::endl;
    std::cout << "║     Hello from Jenga!             ║" << std::endl;
    std::cout << "╚═══════════════════════════════════╝" << std::endl;

    std::cout << "\nCongratulations! Your first Jenga project works!" <<
std::endl;

    if (argc > 1) {
        std::cout << "Arguments: ";
        for (int i = 1; i < argc; ++i) {
            std::cout << argv[i] << " ";
        }
        std::cout << std::endl;
    }

    return 0;
}
```

Étape 3 : Configurer le Build

3.1 Créer le Fichier de Configuration

Créer `hello.jenga` :

```
# Configuration du workspace
with workspace("HelloJenga"):
    # Configurations disponibles
    configurations(["Debug", "Release"])

    # Projet principal
    with project("Hello"):
        # Type de projet
        consoleapp()

        # Langage et standard
        language("C++")
```

```
        cppdialect("C++17")

        # Fichiers sources
        files([
            "src/**.cpp",
            "src/**.h"
        ])

        # Répertoires d'inclusion
        includedirs([
            "include"
        ])

        # Répertoire de sortie
        targetdir("Build/Bin/%{cfg.buildcfg}")

        # Configuration Debug
        with filter("configurations:Debug"):
            defines(["DEBUG"])
            optimize("Off")
            symbols("On")

        # Configuration Release
        with filter("configurations:Release"):
            defines(["NDEBUG"])
            optimize("Full")
            symbols("Off")
```

Explications :

- `workspace("HelloJenga")` : Crée un workspace nommé "HelloJenga"
- `configurations([...])` : Définit Debug et Release
- `with project("Hello")` : Crée un projet "Hello"
- `consoleapp()` : Type = Application console
- `files([...])` : Fichiers à compiler (pattern `**` = récursif)
- `filter(...)` : Applique des settings selon la configuration

🔨 Étape 4 : Compiler le Projet

4.1 Build en Debug (par défaut)

```
jenga build
```

Sortie attendue :

```
```

```
╔═══════════════════════════════════╗
║  Jenga Build System v1.0.2        ║
╚═══════════════════════════════════╝

✓ Workspace 'HelloJenga' loaded with 1 project(s)

============================================================
Building project: Hello
============================================================

ℹ Found 1 source file(s)
✓   [1/1] Compiled: main.cpp
ℹ Linking...
✓ Built: Build/Bin/Debug/Hello.exe

✓ Build completed successfully in 1.2s
```

4.2 Build en Release

```
jenga build --config Release
```

4.3 Rebuild Complet

```
jenga rebuild
```

▶ Étape 5 : Exécuter l'Application

5.1 Exécution Simple

```
jenga run --project Hello
```

Sortie :

```
╔═══════════════════════════════════╗
║     Hello from Jenga!             ║
╚═══════════════════════════════════╝

Congratulations! Your first Jenga project works!
```

5.2 Avec Arguments

```
jenga run --project Hello -- arg1 arg2 arg3
```

Sortie :

```
╔═══════════════════════════════════╗
║     Hello from Jenga!             ║
╚═══════════════════════════════════╝

Congratulations! Your first Jenga project works!
Arguments: arg1 arg2 arg3
```

5.3 Exécution Manuelle

```
# Windows
.\Build\Bin\Debug\Hello.exe

# Linux/macOS
./Build/Bin/Debug/Hello
```

🎓 Étape 6 : Ajouter des Fonctionnalités

6.1 Ajouter un Fichier Header

Créer `include/greeting.h` :

```
#pragma once
#include <string>

namespace greeting {
    std::string getWelcomeMessage();
    void displayBanner();
}
```

Créer `src/greeting.cpp` :

```
#include "greeting.h"
#include <iostream>

namespace greeting {
    std::string getWelcomeMessage() {
        return "Welcome to Jenga Build System!";
```

```
    }

    void displayBanner() {
        std::cout << "╔═══════════════════════════════════════╗" << std::endl;
        std::cout << "║          " << getWelcomeMessage() << "      ║" <<
std::endl;
        std::cout << "╚═══════════════════════════════════════╝" << std::endl;
    }
}
```

Modifier `src/main.cpp` :

```
#include <iostream>
#include "greeting.h"

int main() {
    greeting::displayBanner();
    std::cout << "\nYour project is growing!" << std::endl;
    return 0;
}
```

6.2 Recompiler

```
jenga build
```

Jenga détecte automatiquement les nouveaux fichiers grâce au pattern `src/**.cpp` !

🔧 Étape 7 : Personnalisation

7.1 Changer le Nom de Sortie

```
with project("Hello"):
    consoleapp()

    # Nom personnalisé de l'exécutable
    targetname("MyApp")

    # Sortie : MyApp.exe au lieu de Hello.exe
```

7.2 Ajouter des Defines

```

```

```
with project("Hello"):
    consoleapp()

    # Defines globaux
    defines([
        "APP_NAME=\"HelloJenga\"",
        "APP_VERSION=\"1.0.0\""
    ])
```

Utilisation dans le code :

```
#include <iostream>

int main() {
    std::cout << "App: " << APP_NAME << std::endl;
    std::cout << "Version: " << APP_VERSION << std::endl;
    return 0;
}
```

7.3 Ajouter des Warnings

```
with project("Hello"):
    consoleapp()

    # Activer les warnings
    warnings("Extra")

    # Traiter warnings comme erreurs
    fatalwarnings("On")
```

📊 Étape 8 : Utiliser les Commandes Avancées

8.1 Informations sur le Projet

```
jenga info
```

Sortie :

```
Workspace: HelloJenga
Projects:
  • Hello (ConsoleApp)
    Files: 2
```

```
    Config: Debug
    Output: Build/Bin/Debug/Hello.exe
```

8.2 Nettoyer

```
# Nettoyer les builds
jenga clean

# Nettoyer tout
jenga clean --all
```

8.3 Diagnostic

```
jenga diagnose
```

Vérifie les compilateurs disponibles et la configuration.

🎯 Projet Complet Final

Voici la structure finale :

```
HelloJenga/
├── hello.jenga
├── src/
│   ├── main.cpp
│   └── greeting.cpp
├── include/
│   └── greeting.h
├── Build/
│   ├── Bin/
│   │   ├── Debug/
│   │   │   └── Hello.exe
│   │   └── Release/
│   │       └── Hello.exe
│   └── Obj/
│       └── Debug/
│           └── Hello/
│               ├── main.o
│               └── greeting.o
└── .cjenga/
    └── cbuild.json  # Cache de build
```

☑ Récapitulatif

Vous avez appris à :

- ☑ Créer une structure de projet
- ☑ Écrire un fichier `.jenga`
- ☑ Compiler avec `jenga build`
- ☑ Exécuter avec `jenga run`
- ☑ Ajouter des fichiers
- ☑ Utiliser des configurations (Debug/Release)
- ☑ Personnaliser le build

🎓 Concepts Clés

Workspace

Conteneur de tous vos projets. Un fichier `.jenga` = un workspace.

Project

Une cible de build (exe, lib, dll). Un workspace peut avoir plusieurs projets.

Configuration

Profil de build (Debug, Release, etc.). Change les options de compilation.

Filter

Applique des settings selon des conditions (configuration, plateforme, etc.).

Pattern Files

`src/**.cpp` : Tous les .cpp dans src/ récursivement
`src/*.cpp` : Seulement dans src/ (non récursif)

🚀 Prochaines Étapes

Maintenant que vous maîtrisez les bases, explorez :

Niveau Intermédiaire

- 📓 Tutoriel Partie 1 - Workspaces et projets détaillés
- 📦 Gestion des Projets - Multi-projets
- 🔨 Toolchains - Compiler avec différents toolchains

Niveau Avancé

- 🧪 Tests Unitaires - Ajouter des tests
- 📦 Projets Externes - Réutiliser des bibliothèques

- Multi-Plateforme - Build pour plusieurs OS

Exemples

- Galerie d'Exemples - Projets complets

Problèmes Courants

"Compiler not found"

Solution : Vérifier que le compilateur est installé et dans le PATH.

```
# Vérifier
jenga diagnose

# Windows - Ouvrir Developer Command Prompt
# Linux - Installer build-essential
sudo apt install build-essential
```

"No source files found"

Solution : Vérifier le pattern dans `files([...])`.

```
# Correct
files(["src/**.cpp"])   # Récursif

# Si fichiers à la racine
files(["*.cpp"])
```

"Undefined reference to..."

Solution : Fichier source manquant ou non inclus.

```
# Inclure tous les .cpp
files([
    "src/**.cpp",
    "src/**.c"
])
```

Astuces

Compilation Rapide

```
# Utiliser tous les cores CPU
jenga build -j auto
```

Verbose Mode

```
# Voir les commandes exactes
jenga build --verbose
```

Build Spécifique

```
# Seulement un projet
jenga build --project Hello

# Configuration spécifique
jenga build --config Release
```

Clean Sélectif

```
# Nettoyer seulement un projet
jenga clean --project Hello
```

Ressources

- API Reference - Toutes les fonctions
- Variables - Variables disponibles
- CLI Commands - Commandes complètes
- Troubleshooting - Solutions aux problèmes

Félicitations !
Vous avez créé votre premier projet Jenga avec succès !

Temps passé : ~15 minutes
Compétences acquises : Bases de Jenga
Prochaine étape : Tutoriel Partie 1