

Bologna
Game Farm

Serre dei Giardini Margherita

19 novembre 2025

Programming

Dalla teoria alla realtà del codice

Cosa imparerai durante il tuo primo vero progetto Unreal

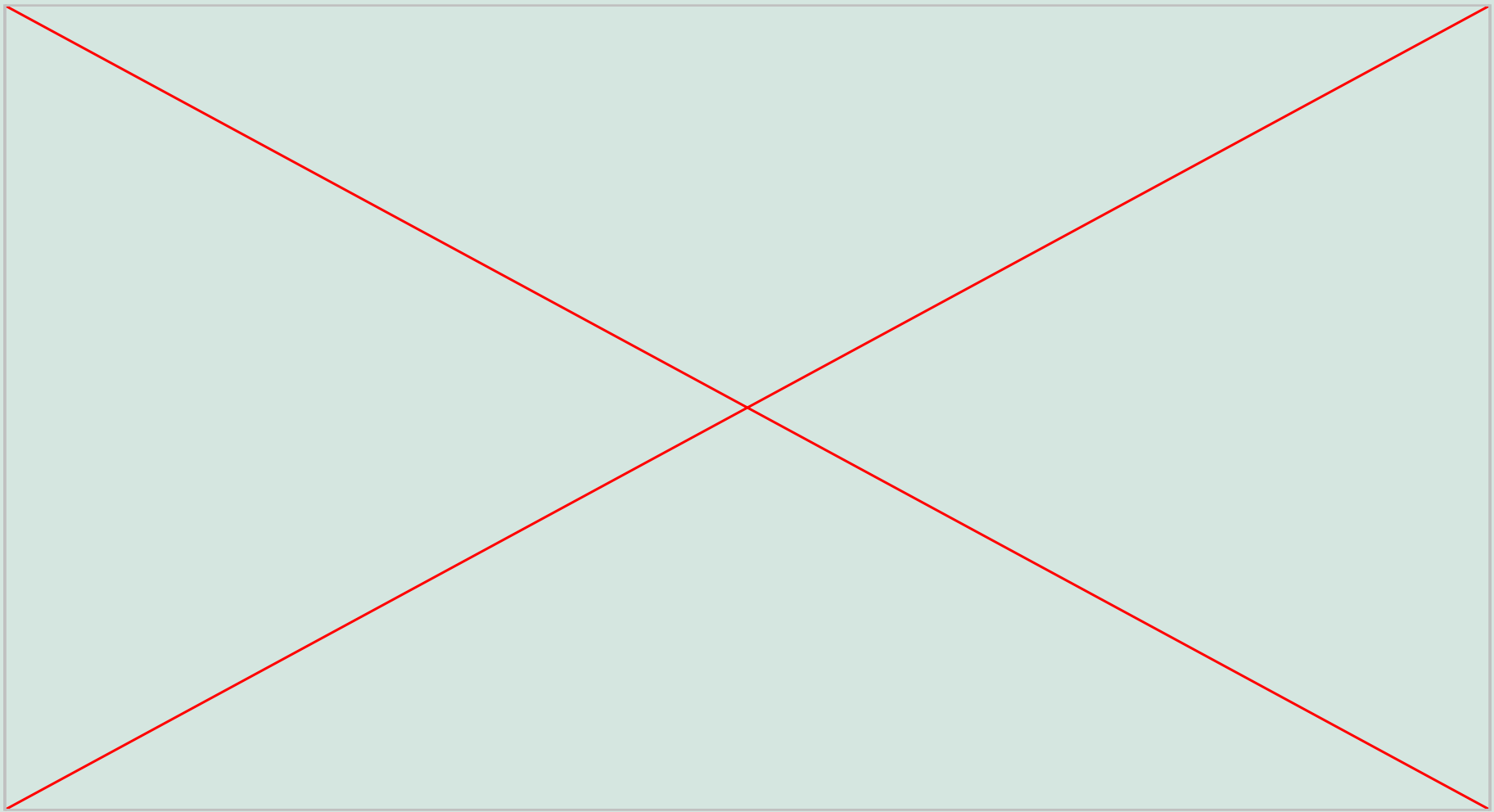
Tomas Bacchiocca

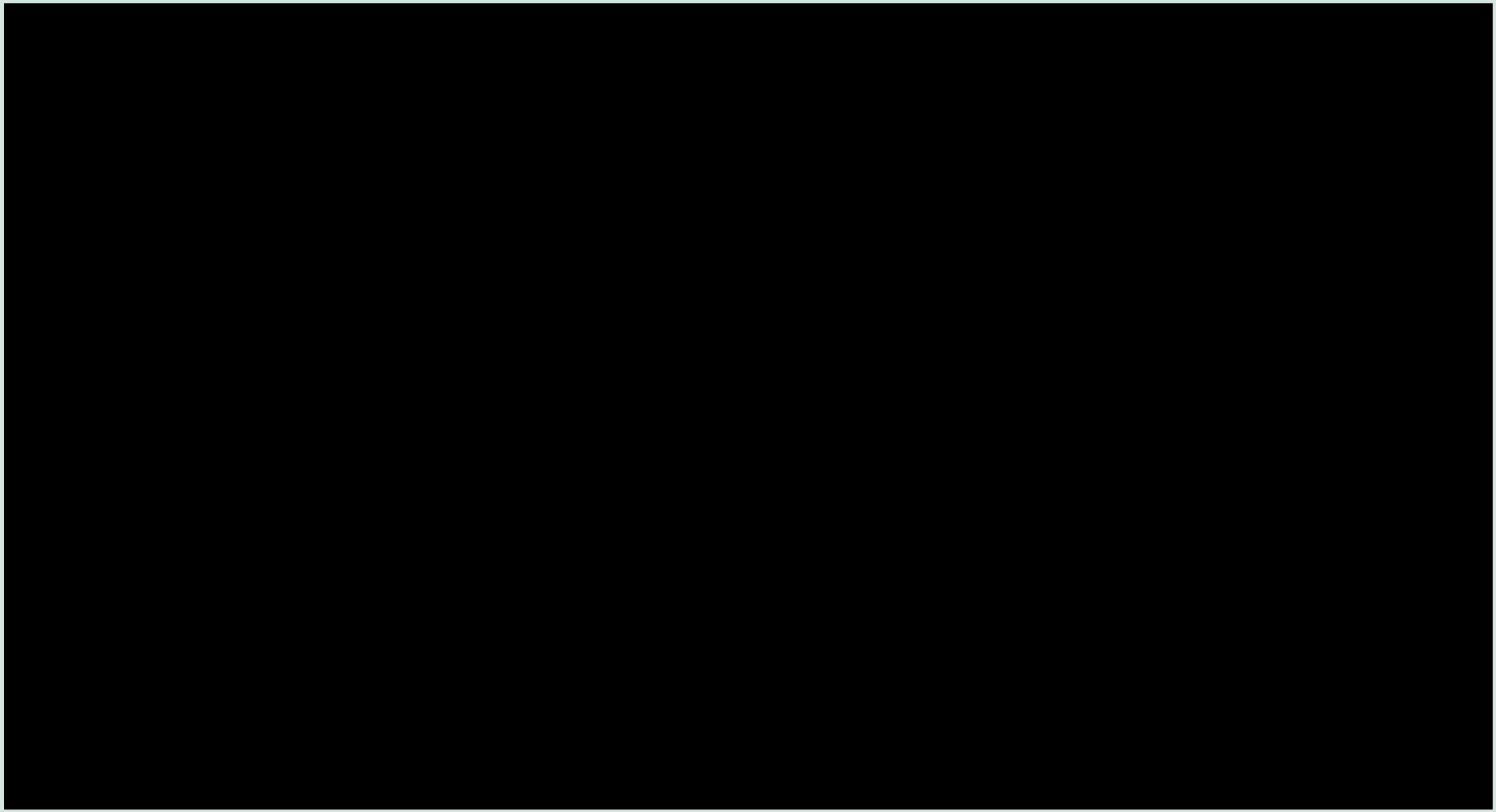


Ministero delle Imprese
e del Made in Italy









Sommario

1

L' Archeologo nel
Codice Altrui

2

La Verità sul
Tuo Codice

3

La Sindrome della
Pagina Bianca

4

La Trappola
Tecnica

5

La Battaglia con
Te Stesso

1

L' Archeologo nel Codice Altrui

Cosa ci si aspetta ad entrare in una codebase già avviata

Idea

- Scriverò nuove feature.
- Creerò il boss più importante del gioco.



Realtà

- Codebase avviata da 3 anni.
- Migliaia di file
- Variabili con nomi strani
- Correggi il bug

SENSO DI LIBERTÀ



PANICO



L'ARCHEOLOGO NEL CODICE ALTRUI

L'importanza del Debugging

- Il Debugging non è un fallimento, è un regalo.
- Metti un breakpoint, guardi i valori, segui il filo.
- Guardi il codice mentre vive.

Il debugger ti insegna come funziona il gioco molto meglio della documentazione (che spesso non c'è).

Il Pericolo del “Codice Zombie”

Pezzi di codice commentati tre anni fa con scritto

//NON TOCCARE o dei **//TODO** ancora da risolvere.

Lo cancello o rompo tutto? Chi si ricorda a che serve...

```
void UFSMPlayerDefaultState::Landed(const FHitResult& Hit)
{
    if (!Player->GetFSM()->IsInStateUsingClass(this->GetClass()))
        return;

    const float LandingSpeed = MovementComponent->Velocity.Z;

    if (LandingSpeed < Player->GetLandingRollSpeedThreshold())
        return;

    Player->LandingState->SetLandingMode(ELandingMode::HardLanding);
    //const ELandingMode mode = LandingSpeed < Player->GetLandingDeathSpeedThreshold() ? ELandingMode::RollLanding : ELandingMode::HardLanding;
    //ELandingMode mode = LastAttemptedRollTimer > 0 ? ELandingMode::RollLanding : ELandingMode::RollLanding; //LastAttemptedRollTimer > 0 ? ELandingMode::RollLanding : ELandingMode::HardLanding; //TODO: Uncomment after demo
    mode = LandingSpeed < Player->GetLandingDeathSpeedThreshold() ? mode : ELandingMode::DeathLanding; //TODO: Uncomment after demo
    //Player->LandingState->SetLandingMode(mode);

    FSM->MoveToStateUsingInstance(Player->LandingState);
}
```

**E' importante tenere pulita la 'casa', perché un giorno tu
sarai il peggior nemico di te stesso!**

2

La Verità sul Tuo Codice

Cosa NON significa scrivere codice

Idea

- Compila + Funziona in gioco = Finito



Realtà

- “Funzionare” è solo il requisito minimo, è la base.

Far funzionare il codice è come scrivere la **brutta copia** di un tema.

Voi la capite, ma se la legge qualcun altro impazzisce.

Pulirla, rinominare le variabili, renderla la **bella copia**.

Deve passare il **QA**.

Solo allora il task è finito.

LA VERITÀ SUL TUO CODICE

La prima Code Review

Ero fiero del mio codice.

Mi è tornato indietro con 20 commenti.

Ci si sente **attaccati**.



La Code Review non è un esame per
bocciarti, è un regalo.

LA VERITÀ SUL TUO CODICE

Scrivere per il “te stesso” del futuro

Scrivere bene significa scrivere una lettera al “te stesso” di tra 6 mesi.



Non ricorderai **NULLA**.

```
//TODO: Remove hardcoded  
EComplexityTier complexity = Data->GetComplexityTier();
```

Ricorda di non lasciare “pezze” nel codice!

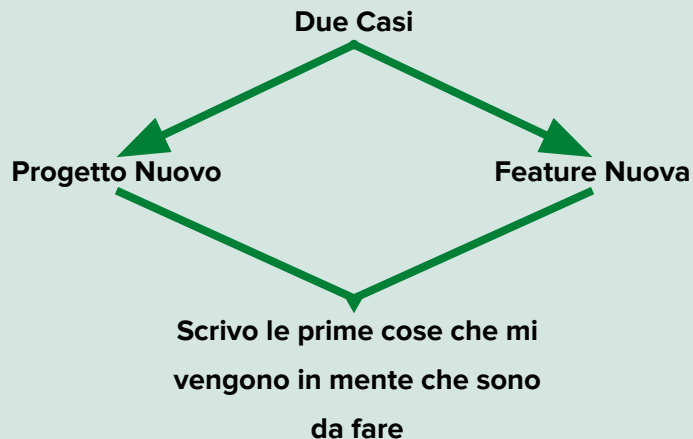
Usate i **//TODO**

La leggibilità è sopravvivenza.

3

La Sindrome della Pagina Bianca

Come NON si inizia un progetto



SBAGLIATO!

LA SINDROME DELLA PAGINA BIANCA

L'organizzazione non è solo burocrazia

E' la **rete di salvezza**.

Non basta dire 'Sul mio PC funziona'.

1. Testo
2. Buildo
3. Provo in Development Build
4. Funziona? Provo in Shipping Build
5. Faccio Git Push



Non sempre bisogna programmare

- Prendere un foglio e scomporre i problemi
- Importare animazioni e inserirci i Notify
- Creare UI
- Creare Documenti Tecnici
- Testing e Debugging
- Fornire gli strumenti ai Designers

- Essere Multiruolo (L'anello di congiunzione tra i reparti)
- Migliorare la Pipeline (Automatizzare il lavoro di altri)
- Ridurre i Colli di Bottiglia (Garantire il flusso di lavoro)

30% Scrittura di Codice

40% Debugging e Test

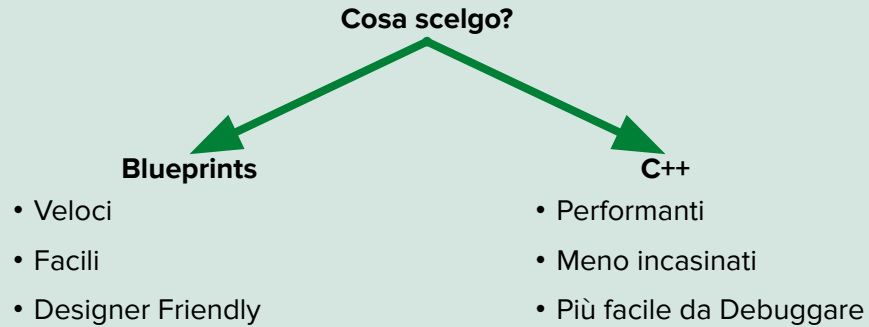
30% Comunicazione e

Tooling

4

La Trappola Tecnica

E' meglio Blueprint o C++?



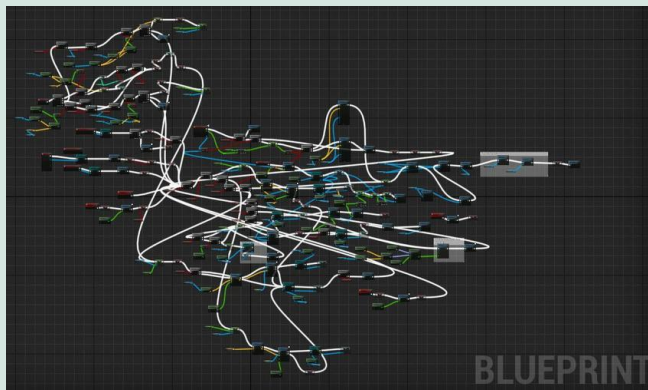
L'approccio vincente è **IBRIDO**

Il Pericolo degli “Spaghetti Blueprint”

Proprio perché i Blueprint sono **facili** e **visivi**, è facilissimo fare **disastri**.

Il problema è che la **libertà visiva** ci rende **pigri**.

Un **Blueprint disordinato** è molto più difficile da debuggare di un C++ scritto male.



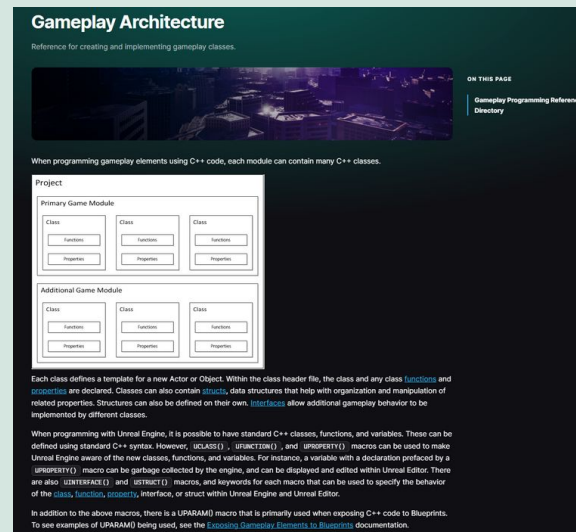
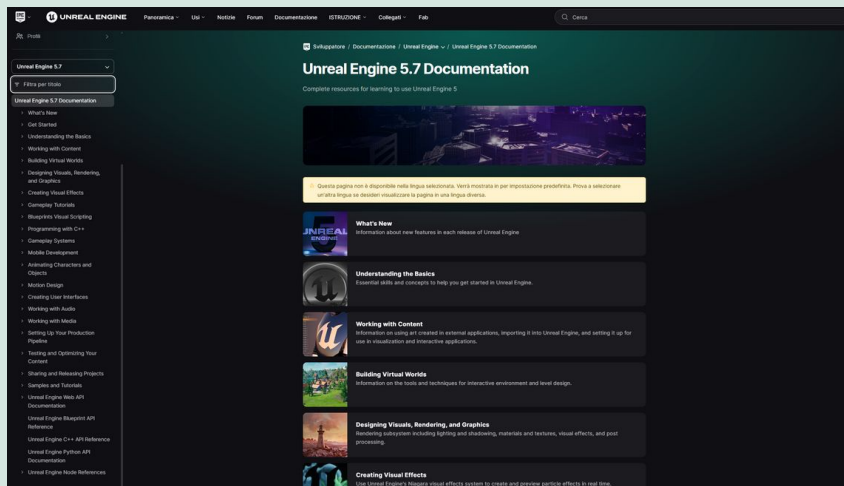
**Trattate i Blueprint con la stessa disciplina del codice:
ordine, commenti, funzioni.**

LA TRAPPOLA TECNICA

Non andare contro all'Engine

L'altra trappola è l'ego.

“Faccio io il sistema di movimento da zero”



Il tuo obiettivo è finire il gioco, non riscrivere il motore.

NON REINVENTATE LA RUOTA

5

La Battaglia con Te Stesso

LA BATTAGLIA CON TE STESSO

La Sindrome dell'Impostore

Sembrano tutti geni.

“Io sono l'unico che non capisce. Mi scopriranno”

Ce l'abbiamo **tutti**, anche i senior.

Accettare di non sapere è il primo passo.

LA BATTAGLIA CON TE STESSO

L'importanza della Comunicazione

E' importante **comunicare**.

Imparate a chiedere, ma fate le domande **giuste**.

Non dite: 'Non funziona' (fa perdere tempo).

Dite: “Sto cercando di fare **X**. Ho provato **A** e **B**. Mi dà l'errore **C**.

Mi sai indicare la direzione?”

Dimostrate impegno e vi dà la risposta in 5 minuti invece che in 5 ore.

LA BATTAGLIA CON TE STESSO

L'Ansia delle Stime

“Quanto ci metti?”

Meglio dare una stima più alta e finire prima, che dare una stima di cui non si è sicuri.

Takeaways

- 1** Sii un Detective:
Debuggare è importante.
- 2** Dalla Brutta alla Bella Copia:
'Funzionare' è solo l'inizio.
- 3** Organizzati:
Il processo (e il test in Shipping)
è la tua sicurezza.
- 4** Usa l'Ibrido:
C++ e Blueprint insieme,
ma in modo ordinato.
- 5** Chiedi (bene):
Non restare bloccato,
comunica.

Grazie a tutti

Contatti

Tomas Bacchiocca

Mail: tomasbacchiocca@gmail.com

Discord: tomas_01