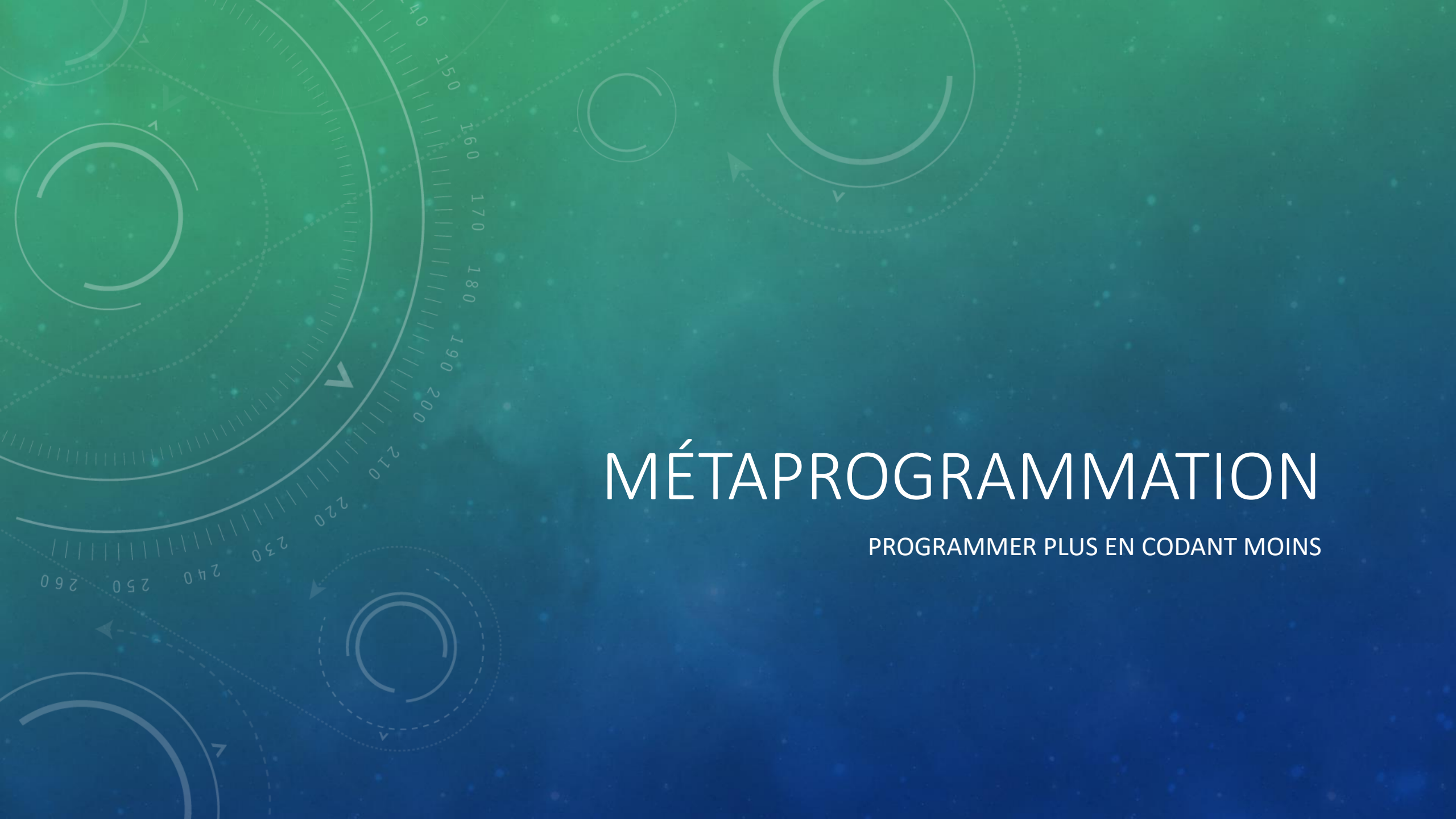


MÉTAPROGRAMMATION

PROGRAMMER PLUS EN CODANT MOINS

PRÉSENTATION PERSONNELLE

Étienne Alby
Programmeur Systèmes / Architecte Logiciel

Contribué aux productions de ces studios :


DÉFINITION

***Métaprogrammation** :*

Désigne l'écriture de programmes qui manipulent des données décrivant elles-mêmes des programmes.

FAIRE TRAVAILLER LA MACHINE

[Computer programming] “should be very fascinating. There need be no real danger of it ever becoming a drudge, for any processes that are quite mechanical may be turned over to the machine itself.”

[La programmation informatique] « devrait être fascinante. Il n’y a aucun danger que ça devienne monotone, car tout procédé un tant soit peu mécanique peut être confié à la machine elle-même. »

-- Alan Turing

CE QUE L'ON PEUT APPELER MÉTAPROGRAMMATION

Avec une définition très (trop) large :

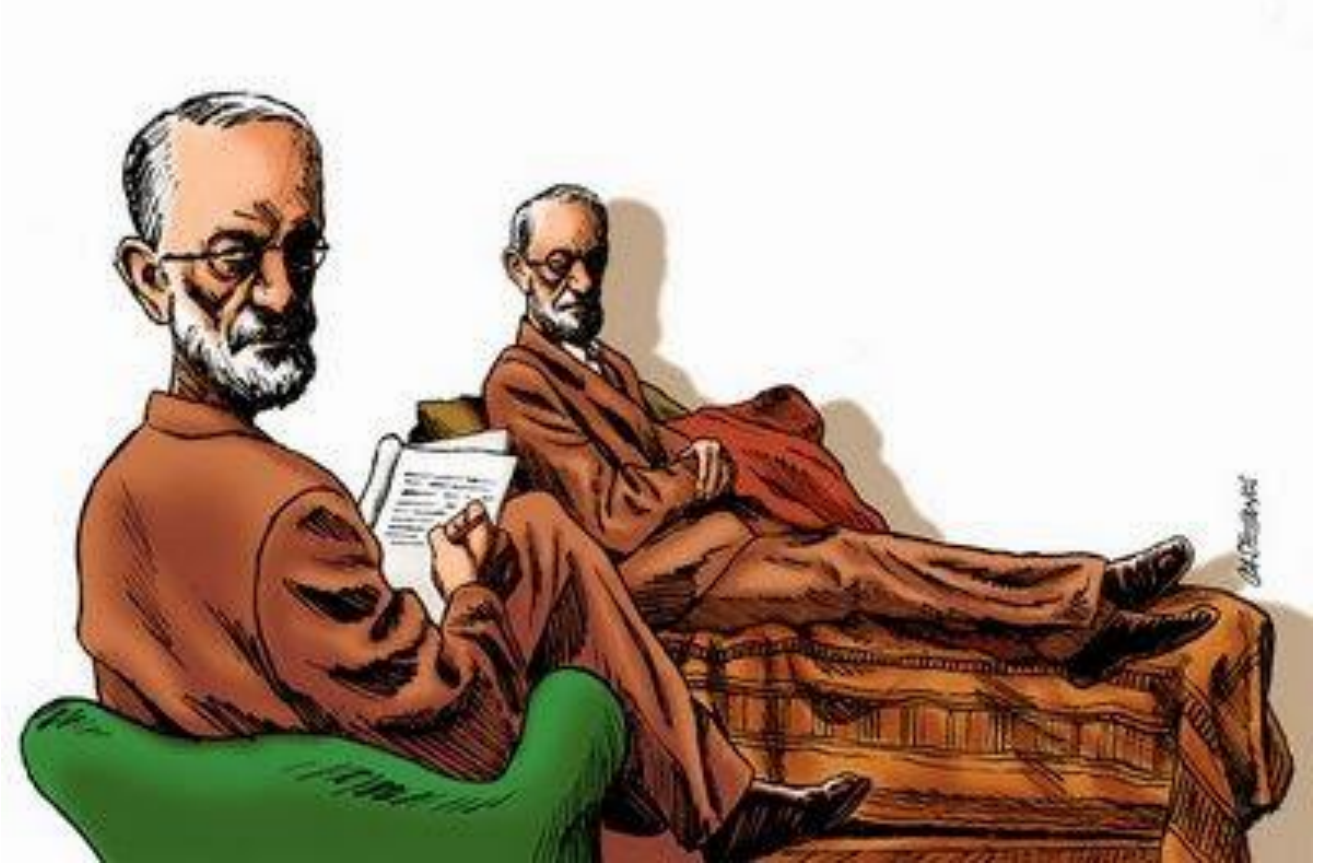
- Préprocesseur et macro en C/C++
- Templates (C++) (pas generics en C#)
- Héritage (POO)
- Completion dans les IDE (Snippets)
- Dernièrement l'usage de LLM (Microsoft Copilot)
- Générateur de code (Source à Source)
- Visual Programming et Optimiseur (compilateur)

DEFINITION DE MOTS CLEFS

- Introspection (Réflexion)
- Template
- Runtime / Static Time

DEFINITION DE MOTS CLEFS

- Introspection

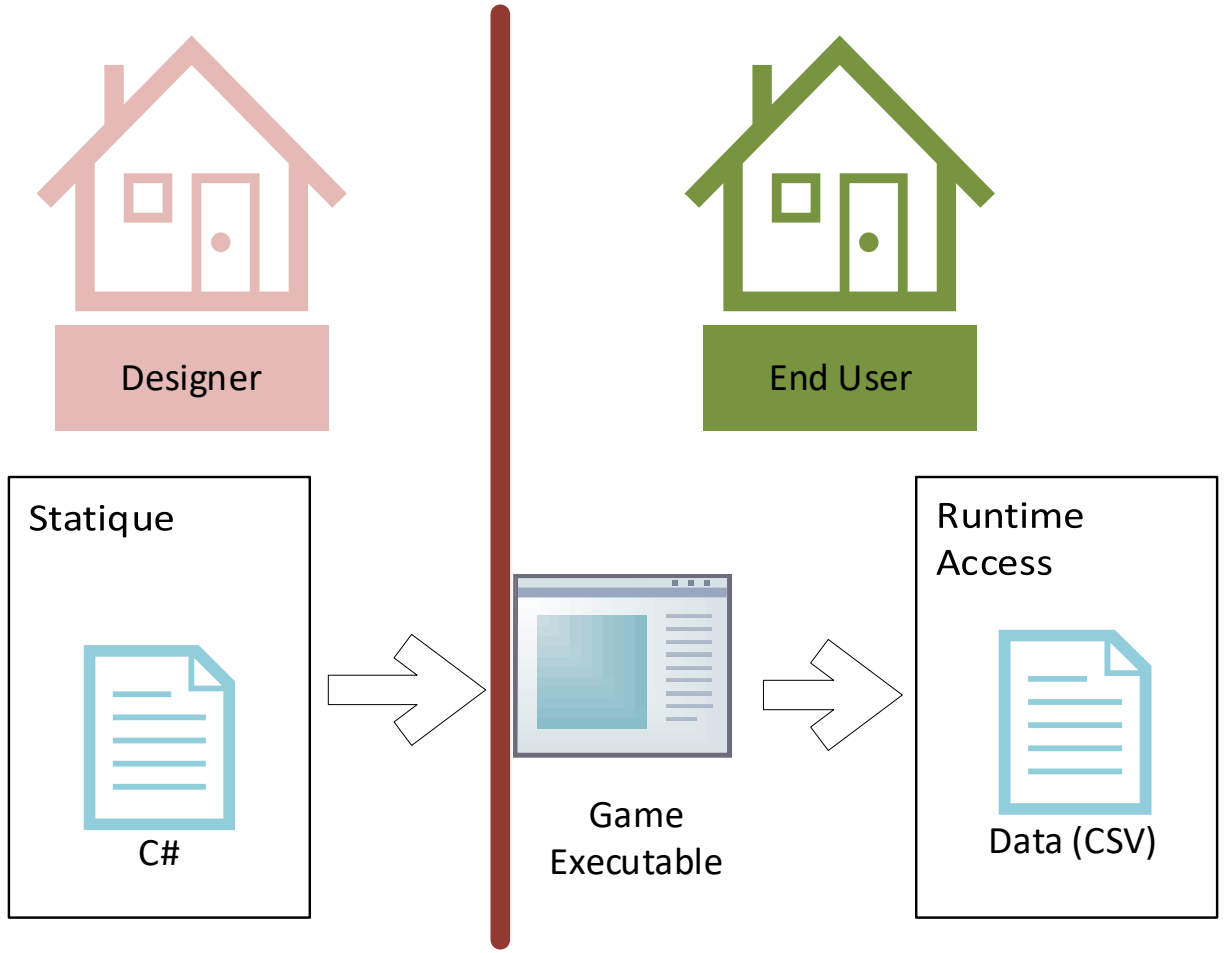


DEFINITION DE MOTS CLEFS

- Template : Moule Industriel

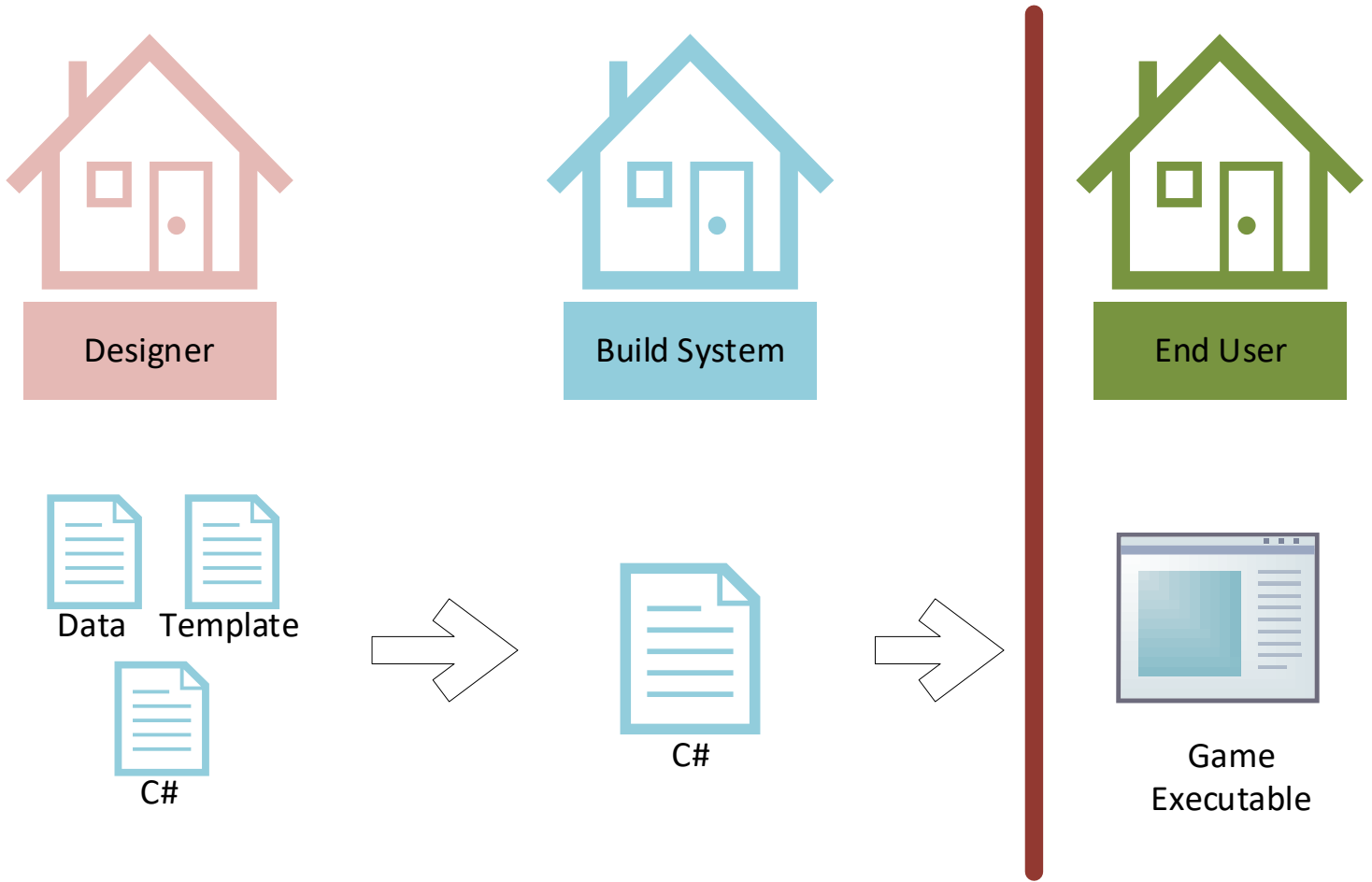
RUNTIME ORIENTED PROCESSING

- Les data sont inconnues
- Problème de sécurité
- Chargement plus long
- Plus complexe de changer le programme en fonction des données



STATIC ORIENTED PROCESSING

- Data déjà préprocessée
- Sécurité accrue
- Rapidité accrue
- Capacité de logique dans le programme accru



INTÉRÊT CONCRET

- DRY (Don’t Repeat Yourself)
- Rapidité d’exécution au Runtime (préprocesseur statique)
- Ajout de capacité à des langages qui n’ont pas d’introspection (SFINAE, …)
- Combinatoire accrue
- Garantie de conformité à la source
- Généricité
- Facilite la maintenance (automatisation d’API)

PROBLÈMES

- Sujet avancé (pas enseigné/mal enseigné)
- Mal intégré dans le langage (templates et preprocessor)
- Système de Build plus complexe (générateur de code Source à Source)
- Manque de connaissance/d’intérêt.

QUAND UTILISER DES TEMPLATES

- En remplacement de la réflexion au runtime pour des données connus à la compilation
- Synchroniser des données entre le code et les assets (bidirectionnel)
- Déléguer des décisions à des designers qui modifient le programme
- Sortir des données du programme (ex: actions possible d'un ennemi)
- Documentation automatique
- Génération d'API dans de nombreux langages
- Spécialisation d'implémentation (extern C, DOM)

EXEMPLES CONCRETS

- Ameliorer la communication entre corps de métiers différents.
- Générer des entrée disponible pour le level designer/ game designer et éditable par lui hors du moteur / hors du code

EXEMPLES CONCRETS - COMBINATOIRE

Définition

Attaque

- Attaque 1
- Attaque 2
- Attaque 3

Vol

- Vol
- Vol Stealth

Résultat

Combinatoire

- Attaque 1
- Attaque 2
- Attaque 3
- Vol
- Vol Stealth
- Attaque 1 & Vol
- Attaque 1 & Vol Stealth
- Attaque 2 & Vol
- Attaque 2 & Vol Stealth
- Attaque 3 & Vol
- Attaque 3 & Vol Stealth

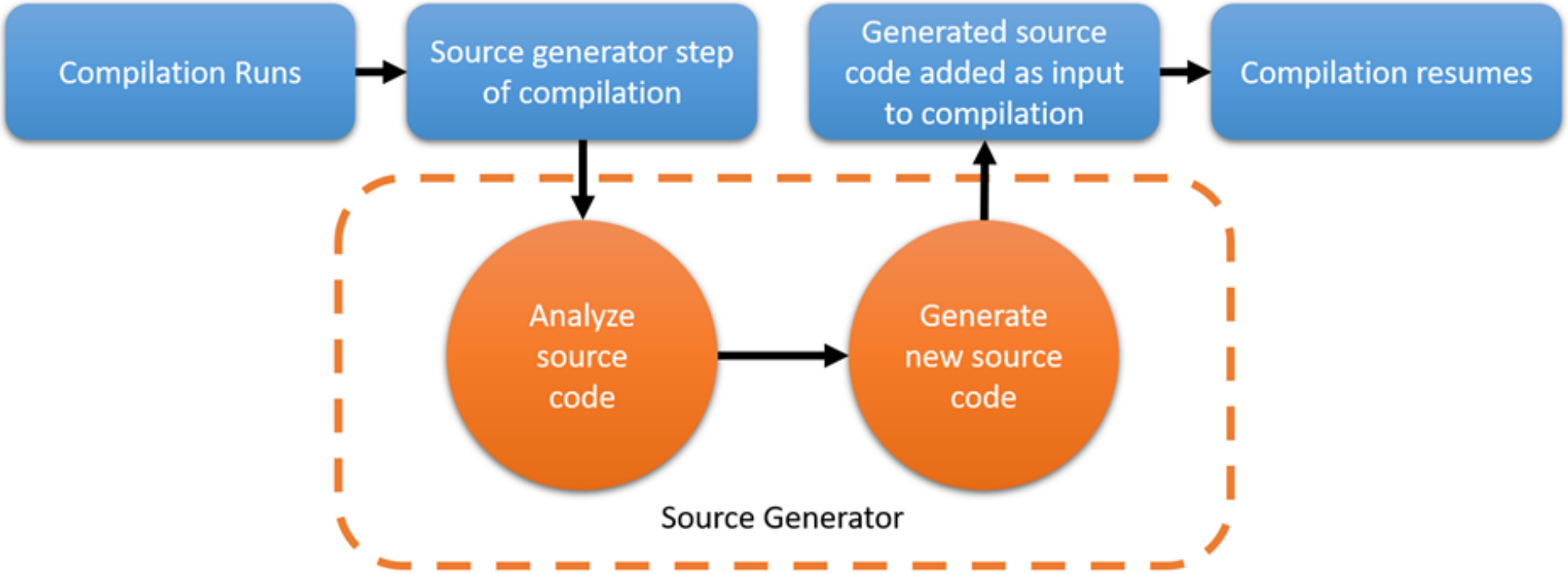
La combinatoire des options peut être préprocessée au lieu d'être calculé au runtime.

- Cela facilite le debug pour le designer qui peut voir les valeurs mergées (potentiellement même les opération mathématiques si les pouvoir sont altérés selon des règles)
- Cela augmente la vitesse de traitement

FOCUS : SOURCE 2 SOURCE

- Le source 2 source (Source to source) : sous-représenté
- Raison principale :
  - Fait rarement partie intégrante du langage (sauf .net Source Generators)
  - Demande un build system avancé
  - Complexifie le pipeline de build (CI)
  - Ignorance de sa disponibilité

.NET SOURCE GENERATORS



CODE GENERATION AVEC C#

- Source Generator est une nouvelle incarnation d'un concept que Microsoft avait présenté depuis des années et qui est déjà intégré à Visual Studio (2005+) : T4
- Text Template Transformation Toolkit: T4 est celui sur lequel nous allons nous concentrer.

TEXT TEMPLATE TRANSFORMATION TOOLKIT

- Supporté dans tous les IDE compatible avec C# et .net :
  - Visual Studio (depuis VS 2005)
  - MonoDevelop
  - JetBrains Rider

CAPACITÉ DES TEMPLATE T4

- Les templates T4 utilisent toutes les fonctionalités du langage C#
- Vous pouvez inclure des Assemblies C# et utiliser le code de ces Assemblies
- Vous pouvez utiliser la réflexion et faire de l'introspection du code C# d'une Assembly
- Cela inclue les classes, les membres de ces classes, les attributs, les enums, etc…
- Vous pouvez utiliser tout type de fichier comme définition de vos données (Assembly DLL, JSON, XML, CSV, etc…
- Il n'y a pas de contrainte sur le fichier en sortie, celui-ci peut être du C++, du C#, du python, un fichier XML, un CSV, etc…

SYNTAXE TEMPLATE T4

- Tout ce qui est dans le fichier .tt se retrouvera en sortie.
- Sauf les directives de templates que l'on va inclure entre « < » et « > »

```
<#@ template hostspecific="false" language="C#" #>
<#@ output extension=".txt" #>
<#int top = 10;

for (int i = 0; i<=top; i++)
{ #>
   The square of <#= i #> is <#= i*i #>
<# } #>
```

TIPS & TRICKS

- Relation **1:1** avec le template (1 fichier **.tt** donne un fichier de sortie)
- T4.FileManager est une extension gratuite qui permet d'avoir **1:N** (plusieurs fichiers sortie)
- Très important : les Assemblies importées doivent être au format **.net standard 2.0**
- Si vous ne pouvez pas lire l'assembly directement (mauvaise version de .net) vous pouvez inclure le fichier dans le template.
- Vous pouvez voir les erreurs de compilation du template en précisant « debug="true" »
- Le projet qui contient les templates doit être un csproj (même si vous l'utilisez pour du C++)

ALTERNATIVES (ET RUNTIME)

- Si vous n'utilisez pas les outils Microsoft sous Windows des alternatives multiplateformes que vous pouvez également inclure pour utiliser au runtime dans vos projets :
- Python : Jinja
- C# : Mono T4
- C++ : Mustache
- Rust : Tera

QUESTIONS

- N’hésitez pas à poser des questions !
- Et à me contacter à l’adresse contact@quantumacy.com

- Les slides de cette présentation seront disponible sur https://quantumacy.com/blog ainsi qu’un article et un lien vers une solution Visual Studio qui présente les patterns présenté dans la présentation

QUANTUMACY

- Studio multimédia fondé en 2020 en Haute Garonne.
- Service disponibles :
  - Architecture de projets complexes
  - Aide au recrutement technique (entretien, sélection de cv, test de programmation)
  - Prototypage technique pour préproduction (ajout fonctionnalité moteurs)
  - Accompagnement technique durant la production (code-review, conseils)
  - Aide à la mise en place d’automatisation de tâches
  - Aide ponctuelle pour optimisation et débogage en fin de production
  - Portage

BONUS SLIDE : HÉRITAGE EN POO

- Il y a trois but différent en programmation orienté objet pour utiliser l'héritage:
  - Spécialisation (Animal (comestible) -> Vache (comestible : true), Animal -> Dog (comestible : false)
  - Substitution (Liskov Substitution) utile pour le pattern Stratégie par exemple
  - Déduplication de code (placer le code dans le parent pour que les enfants l'aient aussi)
- Le premier peut avoir un intérêt mais en général on lui préférera la composition.
- Le deuxième but est le plus utile des trois et le seul que je préconise.
- Le dernier but est à proscrire car il est source de mauvaise pratique (anti-pattern). Il est remplaçable dans tous les cas par un template.