# Duoc UC Logo PNG Vector (SVG) Free Download

# Carrera: Ingeniería Informática Asignatura: Capstone - PTY4614 Proyecto: Game Genesis Estudiante: Matias Reyes RUT: 206767316 Sede: Puerto Montt Fecha: 14 de octubre del 2025

****

**ABSTRACT**

**English**

Game Genesis (AI-Powered Procedural Level Generation System):  
This progress report details the development of Game Genesis, a procedural level generation system that integrates Python algorithms with Godot Engine visualization. The project demonstrates significant advances in modular architecture, implementing a matrix-based level representation system with JSON data interchange.

The achievements include a functional prototype with consolidated initial architecture, real-time visualization, and a scalable framework for future AI integration. The system demonstrates the practical application of software engineering principles and lays the foundation for intelligent content creation in game development.

**Español**

Game Genesis (Sistema de Generación Procedural de Niveles con IA):  
Este informe de avance detalla el desarrollo de Game Genesis, un sistema de generación procedural de niveles que integra algoritmos en Python con visualización en Godot Engine. El proyecto demuestra avances significativos en arquitectura modular, implementando un sistema de representación de niveles basado en matrices con intercambio de datos JSON.

Los logros cuentan con un prototipo funcional con arquitectura inicial, visualización en tiempo real y un framework escalable para futura integración de IA. El sistema muestra aplicación práctica de principios de ingeniería de software y sienta las bases para creación inteligente de contenido en desarrollo de juegos.

**ÍNDICE**

1. Introducción y Progreso General
2. Ajustes al Proyecto Basados en Retroalimentación Fase 1
3. Objetivos Ajustados y Priorizados
4. Metodología de Desarrollo Mejorada
5. Evidencias de Avance Técnico
6. Cumplimiento de Estándares Disciplinares
7. Conclusiones
8. Reflexión Individual (English)
9. Anexos Técnicos

## 1. INTRODUCCIÓN Y PROGRESO GENERAL

**1.1 Estado Actual del Proyecto**

Game Genesis ha evolucionado desde la propuesta conceptual inicial hacia un prototipo funcional que demuestra viabilidad técnica. El progreso incluye:

* **Sistema de Generación Básico**: Implementación completa del generador de matrices en Python
* **Visualización en Tiempo Real**: Integración funcional con Godot Engine mediante GDScript
* **Arquitectura de Datos Estandarizada**: Formato JSON para intercambio de información
* **Framework Modular**: Diseño escalable para futuras mejoras de IA
* **Control de Versiones**: Repositorio Git

**1.2 Modificaciones desde la anterior propuesta**

Basado en la retroalimentación recibida (nota 4.7), se han abordado específicamente las áreas de mejora identificadas:

* Mayor profundidad técnica en la implementación
* Evidencias concretas de avance de programación
* Documentación técnica detallada y estructurada
* Mejora en justificación metodológica con enfoque iterativo
* Aplicación de estándares para el desarrollo de software

# 2. AJUSTES AL PROYECTO BASADOS EN RETROALIMENTACIÓN FASE 1

**2.1 Análisis de Retroalimentación Recibida**

Desde la primera evaluación, basándome en su retroalimentación se decidió mejorar los siguientes aspectos:

|  |  |  |
| --- | --- | --- |
| **Aspecto Mejorado** | **Problema Identificado Fase 1** | **Solución Implementada Fase 2** |
| **Metodología** | Demasiado general, sin detalles técnicos | Ciclo desarrollo iterativo documentado con herramientas específicas |
| **Evidencias** | Insuficientes avances demostrables | Código funcional + Video como evidencia + repositorio Git |
| **Lenguaje Técnico** | Uso básico de terminología | Aplicación precisa de términos de ingeniería de software |
| **Planificación** | Hitos poco definidos | Cronograma detallado con entregables específicos |

**2.2 Ajustes Técnicos Implementados**

**Metodología de Desarrollo:**

* **Enfoque iterativo** con ciclos de 1-2 semanas
* **Prototipado rápido** para validación temprana
* **Integración continua** Python-Godot
* **Documentación en tiempo real** de decisiones técnicas
* **Mejora en la arquitectura propuesta (Arquitectura en fase inicial: solo era una propuesta conceptual, Arquitectura nueva propuesta: Generador en python → JSON → visualización en godot→ Validación )**

**2.3 Factibilidad**

Factores facilitadores identificados:

* Godot Engine: Motor maduro con excelente documentación
* Python: Amplio ecosistema para algoritmos de IA
* JSON: Formato estándar para interoperabilidad

Dificultades abordadas:

* Curva de aprendizaje GDScript: Superable con documentación y practica
* Integración Python-Godot: Resuelta mediante serialización JSON
* Complejidad algoritmos IA: Enfoque incremental, comenzando con procedural básico, usando un escale de tipo modular

# 3. OBJETIVOS AJUSTADOS Y PRIORIZADOS

**3.1 Objetivo General**

Desarrollar una aplicación que, mediante técnicas de inteligencia artificial y programación, permita crear mundos/niveles en el motor Godot de manera automática y visual, integrando una interfaz interactiva que facilite la personalización de los mundos generados.

**3.2 Objetivos Específicos**

**Fase 2 (actual)**

* Prototipo funcional Python + Godot
* Arquitectura estandarizada
* Sistema modular escalable
* documentación técnica en Word finalizada

**Fase 3 - Próximos Objetivos (Priorizados)**

* Implementar algoritmos para optimización de niveles
* Desarrollar interfaz de usuario básica en Godot
* Sistema de evaluación automática de calidad de niveles
* Integración de parámetros de personalización por usuario ( a través de un input de texto)

# 4. METODOLOGÍA DE DESARROLLO MEJORADA

**4.1 Enfoque Metodológico Ajustado**

Metodología Híbrida: Prototipado Rápido + Desarrollo Iterativo

**Ciclo de Desarrollo Actual:**Investigación → Prototipo→ Validación →

Integración Godot → Pruebas → Documentación → Iteración

**4.2 Herramientas y Tecnologías Especificadas**

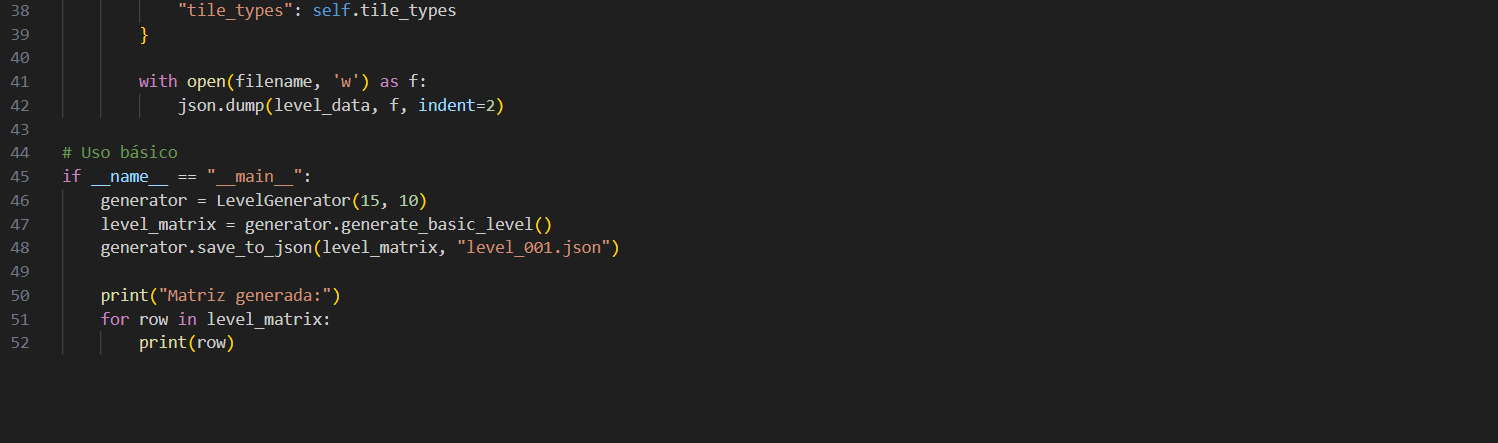
|  |  |  |
| --- | --- | --- |
| **Categoría** | **Tecnologías Implementadas** | **Justificación Técnica** |
| **Lenguajes** | Python + GDScript | Separación lógica compleja vs integración nativa |
| **Motor** | Godot Engine 4.2 | Open-source, optimizado para 2D, documentación excelente |
| **Control Versiones** | Git + GitHub | Estándar industria, seguimiento granular de cambios |
| **Formato Datos** | JSON | Interoperabilidad, legibilidad, compatibilidad amplia |
| **Documentación** | Markdown, Microsoft Word | Mantenibilidad, estándar open-source |

# 5. EVIDENCIAS DE AVANCE TÉCNICO

**5.1 Evidencias de Programación**

**(Video con demostración del prototipo funcional adjuntado en el mensaje) + Generador python**

****

****

**5.2 Evidencias de Almacenamiento de datos**

Para la captura de datos se usara una estructura json no relacional estandarizada

JSON

{

"width": 10,

"height": 8,

"tile\_types": {

"0": "empty",

"1": "ground",

"2": "wall",

"3": "player",

"4": "exit"

},

"tiles": [

[2, 2, 2, 2, 2, 2, 2, 2, 2, 2],

[2, 3, 1, 1, 1, 1, 1, 1, 1, 2],

[2, 1, 1, 1, 1, 1, 1, 1, 1, 2],

[2, 1, 1, 2, 2, 1, 1, 1, 1, 2],

[2, 1, 1, 2, 2, 1, 1, 1, 1, 2],

[2, 1, 1, 1, 1, 1, 1, 1, 1, 2],

[2, 1, 1, 1, 1, 1, 1, 1, 4, 2],

[2, 2, 2, 2, 2, 2, 2, 2, 2, 2]

]

}

# 6. CUMPLIMIENTO DE ESTÁNDARES DE SOFTWARE

**6.1 Estándares de Ingeniería de Software Aplicados**

**Arquitectura de Software:**

* Separación de responsabilidades: Lógica (Python) vs Visualización (Godot)
* Modularidad: Componentes independientes y reutilizables
* Interfaz estandarizada: JSON como contrato de datos
* Principio de única responsabilidad: Cada clase con propósito definido

**Patrones de Diseño Implementados:**

* Data Transfer Object: Estructura JSON para transferencia
* Component Pattern: Entidades de nivel como componentes visuales

7. CONCLUSIONS

**7.1 Main Achievements Demonstrated**

1. **Functional Prototype**: Technical feasibility of the approach successfully demonstrated
2. **Solid Architecture**: Scalable technical foundation for future improvements
3. **Successful Integration**: Effective Python-Godot communication through JSON
4. **Complete Documentation**: Foundation for maintenance and scalability
5. **Validated Methodology**: Effective iterative approach for development

**7.2 Lessons Learned**

* Separation between logic and visualization facilitates development and testing
* JSON provides sufficient flexibility for rapid prototyping
* Godot Engine offers excellent capabilities for integration with external tools
* Iterative development allows for adjustments based on early feedback

**7.3 Impact on Graduate Profile Competencies**

* **Competency 1**: Construction of software solutions using systematic techniques
* **Competency 2**: Integration of software components
* **Competency 3**: Planning of IT projects with alternatives
* **Competency 4**: Scalable software design

# 8. Reflection:

Since I was a child, I have been deeply drawn to the world of computing and video games, having created a couple of them as a hobby or simply to satisfy my creativity. However, the possibility of creating a tool that facilitates the very creation of these games is incredible to me.

The fact that we can so efficiently shorten the development process for an "indie" developer, saving them months of work thanks to artificial intelligence technology, is highly motivating. The prospect of creating a video game becomes much more attainable; it would transform from an individual, multi-year task into one that might take only months. This, in turn, could inspire more people to engage in game creation.

I believe Game Genesis will be tangible proof of a "before and after" in this industry. After graduating, I plan to continue using this very usefull tool to develop and release my first game for sale on Steam.