Universidad Católica San Pablo (UCSP)
Escuela Profesional de
Ciencia de la Computación
SILABO

CS111. Programación de Video Juegos (Obligatorio)

2023-I

1. Información general		
1.1 Escuela	:	Ciencia de la Computación
1.2 Curso	:	CS111. Programación de Video Juegos
1.3 Semestre	:	1^{er} Semestre.
1.4 Prerrequisitos	:	Ninguno
1.5 Condición	:	Obligatorio
1.6 Modalidad de aprendizaje	:	Presencial
1.7 horas	:	2 HT; 4 HP;
1.8 Créditos	:	4
1.9 Plan	:	Plan Curricular 2016

2. Profesores

Titular

- Graciela Lecireth Meza Lovón <gmezal@ucsp.edu.pe>
  - Doctor en Ciencia de la Computación, Universidad Nacional San Agustín, Perú, 2016.
  - Master en Ciencia de la Computación, UFMS-MS, Brasil, 2007.
- Kelly Vizconde la Motta <kvizconde@ucsp.edu.pe>
  - Master en Mag. Ciencia de la Computación, Universidad Católica San Pablo, Perú, 2019.

Laboratorio

- Yessenia Deysi Yari Ramos <ydyari@ucsp.edu.pe>
  - Master en Ciencias de la Computación, UFRGS, Brasil, 2011.

3. Fundamentación del curso

Este es el primer curso en la secuencia de los cursos introductorios a la Ciencia de la Computación. En este curso se pretende cubrir los conceptos señalados por la Computing Curricula IEEE-CS/ACM 2013. La programación es uno de los pilares de la Ciencia de la Computación; cualquier profesional del Área, necesitará programar para concretizar sus modelos y propuestas. Este curso introdución a los participantes en los conceptos fundamentales de este arte. Lo tópicos incluyen tipos de datos, estructuras de control, funciones, listas, recursividad y la mecánica de la ejecución, prueba y depuración.

4. Resumen

1. Historia de los Lenguajes de Programación 2. Conceptos Fundamentales de Programación 3. Algoritmos y Estructuras de Datos fundamentales 4. Programación orientada a objetos 5. Programación de Video Juegos

5. Objetivos Generales

- Introducir los conceptos fundamentales de programación.
- Desarrollar su capacidad de abstracción utilizar un lenguaje de programación.

6. Contribución a los resultados (*Outcomes*)

Esta disciplina contribuye al logro de los siguientes resultados de la carrera:

1) S.O. Analizar un problema computacional complejo y aplicar los principios computacionales y otras disciplinas relevantes para identificar soluciones. (**Usar**)

2) S.O. Diseñar, implementar y evaluar una solución basada en computación para cumplir con un conjunto determinado de requisitos computacionales en el contexto de las disciplinas del programa. (**Usar**)

6) S.O. Aplicar la teoría de la computación y los fundamentos del desarrollo de software para producir soluciones basadas en computación. . (**Usar**)

7. Contenido

UNIDAD 1: Historia de los Lenguajes de Programación (5)

Competencias:

Contenido	**Objetivos Generales**
• Perspectiva histórica de los lenguajes de programación. • Conceptos de Programación tradicionales.	• Identificar importantes tendencias en la historia del campo de la computación [Familiarizarse] • Discutir el contexto histórico de los paradigmas de diversos lenguajes de programación [Familiarizarse]

Lecturas: Brookshear and Brylow (2019)

UNIDAD 2: Conceptos Fundamentales de Programación (9)	
Competencias: 1	
Contenido	**Objetivos Generales**
• Sintaxis y semántica básica de un lenguaje de alto nivel. • Variables y tipos de datos primitivos (ej., numeros, caracteres, booleanos) • Expresiones y asignaciones. • Operaciones básicas I/O. • Estructuras de control condicional e iterativas. • Funciones definidas por el usuario. • Paso de funciones y parámetros. • Concepto de recursividad.	• Analiza y explica el comportamiento de programas simples que involucran estructuras fundamentales de programación variables, expresiones, asignaciones, E/S, estructuras de control, funciones, paso de parámetros, y recursividad [Evaluar] • Identifica y describe el uso de tipos de datos primitivos [Familiarizarse] • Escribe programas que usan tipos de datos primitivos [Usar] • Modifica y expande programas cortos que usen estructuras de control condicionales e iterativas así como funciones [Usar] • Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar] • Escoje estructuras de condición y repetición adecuadas para una tarea de programación dada [Familiarizarse] • Describe el concepto de recursividad y da ejemplos de su uso [Evaluar] • Identifica el caso base y el caso general de un problema basado en recursividad [Familiarizarse]
Lecturas: Guttag (2013), Zelle (2010)	

UNIDAD 3: Algoritmos y Estructuras de Datos fundamentales (8)	
Competencias: 1	
Contenido	**Objetivos Generales**
• Algoritmos numéricos simples, tales como el cálculo de la media de una lista de números, encontrar el mínimo y máximo. • Algoritmos de ordenamiento de peor caso cuadrático (selección, inserción) • Algoritmos de ordenamiento con peor caso o caso promedio en O(N lg N) • Algoritmos de búsqueda secuencial y binaria.	• Implementar algoritmos numéricos básicos [Usar] • Implementar algoritmos de busqueda simple y explicar las diferencias en sus tiempos de complejidad [Evaluar] • Ser capaz de implementar algoritmos de ordenamiento comunes cuádraticos y O(N log N) [Usar] • Discutir el tiempo de ejecución y eficiencia de memoria de los principales algoritmos de ordenamiento y búsqueda. • Discutir factores otros que no sean eficiencia computacional que influyan en la elección de algoritmos, tales como tiempo de programación, mantenibilidad, y el uso de patrones específicos de la aplicación en los datos de entrada [Familiarizarse] • Demostrar habilidad para evaluar algoritmos, para seleccionar de un rango de posibles opciones, para proveer una justificación por esa selección,y para implementar el algoritmo en un contexto en específico [Evaluar] • Trazar y/o implementar un algoritmo de comparación de string [Usar]
Lecturas: Guttag (2013), Zelle (2010)	

UNIDAD 4: Programación orientada a objetos (4)	
Competencias: 1	
Contenido	**Objetivos Generales**
• Lenguajes orientados a objetos para la encapsulación: Privacidad y visibilidad de miembros de la clase. • Definición de las categorías, campos, métodos y constructores. • Subclases y herencia. • Asignación dinámica: definición de método de llamada.	• Diseñar e implementar una clase [Usar] • Usar subclase para diseñar una jerarquía simple de clases que permita al código ser reusable por diferentes subclases [Familiarizarse] • Comparar y contrastar (1) el enfoque procedurar/funcional- definiendo una función por cada operación con el cuerdo de la función proporcionando un caso por cada variación de dato - y (2) el enfoque orientado a objetos - definiendo una clase por cada variación de dato con la definición de la clase proporcionando un método por cada operación. Entender ambos enfoques como una definición de variaciones y operaciones de una matriz [Familiarizarse] • Explicar la relación entre la herencia orientada a objetos (codigo compartido y *overriding*) y subtipificación (la idea de un subtipo es ser utilizable en un contexto en el que espera al supertipo) [Familiarizarse] • Usar mecanismos de encapsulación orientada a objetos [Familiarizarse].
Lecturas: Guttag (2013), Zelle (2010)	

UNIDAD 5: Programación de Video Juegos ()	
Competencias: 1	
Contenido	**Objetivos Generales**
• Uso de una librería de video juegos.	• Aplicar los fundamentos y los conceptos de lenguajes de programación para el desarrollo de un video juego simple. [Usar]
Lecturas: sweigart2012making	

8. Metodología

1. El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.
2. El profesor del curso presentará demostraciones para fundamentar clases teóricas.
3. El profesor y los alumnos realizarán prácticas
4. Los alumnos deberán asistir a clase habiendo leído lo que el profesor va a presentar. De esta manera se facilitará la comprensión y los estudiantes estarán en mejores condiciones de hacer consultas en clase.

9. Evaluar Sesiones Teóricas:
Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

Sesiones Prácticas:
Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

Sistema de Evaluación:
La nota final se obtiene a través de:

EVALUACIONES PERMANENTES	EVALUACIONES
Evaluación Permanente 1 : 20 % **Evaluación Permanente 2** : 25 %	**Evaluación Parcial** : 30 % **Evaluación Final** : 25 %
45%	55%

Donde:

Evaluación Permanente: Comprende trabajos grupales, participación activa en clase, test de ejercicios.

- Permanente 1 (Semanas 1 - 9)
- Permanente 2 (Semanas 10 - 17)

Para aprobar el curso, el alumno debe obtener 11.5 o más en la nota final.

References

Brookshear, J. Glenn and Dennis Brylow (2019). *Computer Science: An Overview*. Ed. by PEARSON. Global Edition. Pearson. ISBN: 1292263423.

Guttag, John V (2013). . *Introduction To Computation And Programming Using Python*. MIT Press.

Zelle, John (2010). *Python Programming: An Introduction to Computer Science*. Franklin, Beedle & Associates Inc.