**1. Introduzione**

* Nome del progetto: Fall Simulator
* Obiettivo del progetto: Il progetto ha l’obiettivo di simulare il moto di caduta libera lungo curve di corpi su cui agisce la sola gravità; calcolando i parametri del moto tra cui di particolare interesse il tempo totale di caduta.
* Motivazione e contesto: La questione che ha dato origine a Fall Simulator è il Problema della Brachistocrona (Johann Bernoulli, 1696),uno storico problema di ottimizzazione che richiede di trovare la curva che minimizza il tempo per percorrere lo spazio tra due punti che un corpo in caduta sotto l'effetto della gravità impiega per percorrere lo spazio tra due punti.Il problema si risolve matematicamente con la teoria del Calcolo delle Variazioni, ottenendo che la soluzione ottimale è la curva cicloide, ovvero la traiettoria tracciata dal moto di un punto di una circonferenza che rotola. Il progetto mira a mostrare accuratamente la meccanica del moto in questione e analizzarne i parametri, grazie a calcoli numerici precisi che ricreano il modello del fenomeno fisico.
* Stakeholder: Persone interessate all' analisi simulativa di un modello di fenomeno fisico legato a uno storico problema di fisica matematica. Figure educative che vogliono insegnare il Problema o il fenomeno fisico in modo visivo e chiaro.
* Sintesi del progetto: Il software permette di selezionare due punti nello spazio (con la possibilità di includere punti intermedi) e generare una curva che passa per questi punti. Successivamente, anima una sfera lungo questa curva per studiare il suo comportamento. FallSimulator propone inoltre curve preimpostate, come parabole, rette e cicloidi, per analisi rapide.

**2. Modello di Processo**

Per sviluppare Fall Simulator viene seguito il modello di ciclo di vita della prototipazione evolutiva, addentrandosi presto nei dettagli e nelle problematiche per rivelare fin da subito le questione che richiedono il maggiore sforzo, le quali verranno affrontate in modo più pianificato, con l'obbiettivo di aggiustare e fare evolvere il prototipo verso il software finale. L'approccio allo sviluppo è di tipo Agile piuttosto che basato sulla pianificazione, in particolare nella gestione del processo.

**3. Organizzazione del progetto**

Il team di sviluppo del progetto FallSimulator è composto da due sviluppatori, Marco Locatelli e Nava Nicolò, che collaborano su tutti gli aspetti del progetto. Nicolò ha il focus sull' organizzazione del processo, la documentazione la gestione del progetto,e il testing del sistema; mentre Marco è focalizzato sull'analisi del dominio, la progettazione, modellazione e implementazione del software. Entrambi collaborano e revisionano il lavoro dell'altro.

* Revisore: Il progetto verrà valutato dal professore Gargantini, docente di ingegneria del software.
* Strumenti e comunicazione: Il team utilizza GitHub per il controllo di versione e la gestione del codice. La maggior parte degli incontri e dei confronti avviene di persona, facilitando una comunicazione diretta e collaborativa, oltre a mezzi di comunicazione rapidi e informali.

**4. Standard, linee guida e procedure**

* Convenzioni di codifica: Il team segue le convenzioni ufficiali di Oracle per Java, incluse regole come:
* Classi in PascalCase.
* Variabili e metodi in camelCase.
* Costanti in MAIUSCOLO\_CON\_SOTTOLINEATURE.
* Strumenti di sviluppo: Il progetto utilizza Maven per la gestione delle dipendenze e il controllo del ciclo di build. Il repository è gestito su GitHub per il controllo di versione.
* Gestione della documentazione: Anche se la documentazione non è un focus primario, il team utilizza Javadoc per mantenere una documentazione minima del codice. Nicolò si occupa principalmente della documentazione di progetto.
* Code review: Le modifiche al codice vengono sottoposte a revisione tramite GitHub, per garantire la qualità del codice e prevenire errori.
* Controllo di versione: Il team utilizza GitHub per gestire le versioni del progetto e tenere traccia delle modifiche.

**5. Attività di gestione**

* Monitoraggio del progresso: Il team utilizza Daily Stand-up Meeting per confrontarsi quotidianamente sui progressi e identificare eventuali ostacoli. Inoltre, dopo ogni Sprint, viene svolta una Sprint Review per esaminare il lavoro completato e pianificare le attività successive.
* Stabilire le priorità: Le funzionalità principali, come l'animazione e il calcolo del tempo, hanno la priorità. Una volta completate, il team procede con l'aggiunta di altre funzionalità, come le curve preimpostate e le ottimizzazioni.
* Gestione dei problemi: I problemi vengono risolti attraverso una comunicazione diretta all'interno del team. Se il problema non può essere risolto internamente, viene richiesto aiuto al professore Gargantini.

**6. Rischi**

* Difficoltà nei calcoli matematici per le curve: È previsto il rischio che il calcolo della circonferenza possa presentare difficoltà, poiché richiede non solo i punti di passaggio ma anche il raggio. Questo problema potrebbe rendere più complessa la generazione della curva. Il team ha pianificato di risolvere il problema affrontandolo prima matematicamente e poi applicandolo all'interno dell'applicazione.
* Problemi di animazione: C'è il rischio che l'animazione possa non essere supportata correttamente o risultare lenta, influenzando negativamente le prestazioni del simulatore. Per mitigare questo rischio, è stato previsto del tempo extra per test approfonditi e ottimizzazioni delle prestazioni.

**7. Personale**

* Sviluppo del codice: Questa attività è svolta principalmente da Marco Locatelli, che si occupa della scrittura del codice per la generazione delle curve, il calcolo del tempo "t", e l'implementazione delle animazioni.
* Analisi matematica: Anche l'analisi matematica necessaria per implementare correttamente le curve è principalmente gestita da Marco, con l'aiuto di Nicolò nei momenti necessari.
* Gestione del progetto e documentazione: Nicolò Nava si concentra principalmente sulla gestione del project plan e della documentazione, anche se entrambi i membri del team contribuiscono quando necessario.
* Testing e ottimizzazione: Entrambi i membri del team collaborano per la fase di test e ottimizzazione, con l'obiettivo di garantire un’animazione fluida e corretta, oltre a un calcolo preciso del tempo "t".

**8. Metodi e tecniche**

* Progettazione del software:La progettazione del software è stata realizzata utilizzando diversi diagrammi, tra cui:
  + Diagramma di classi
  + Diagramma d'uso
  + Diagramma di stati  
    Questi diagrammi sono stati creati utilizzando il tool Papyrus.
* Controllo della versione: Il team utilizza GitHub per gestire il controllo delle versioni del software, tenere traccia delle modifiche e collaborare efficacemente.
* Test e verifica: I test vengono eseguiti verificando i risultati prodotti dall'applicazione attraverso calcoli manuali, per assicurarsi che l'app faccia i conti correttamente, specialmente per quanto riguarda il calcolo delle curve e del tempo "t".

**9. Garanzia di qualità**

* Test frequenti: I test vengono eseguiti regolarmente per garantire la correttezza del software. In alcuni casi, vengono sviluppati piccoli programmi per eseguire test specifici su passaggi critici. I calcoli matematici sono verificati manualmente, mentre l'aspetto grafico viene confrontato con altre applicazioni, come GeoGebra.
* Criteri di qualità: Il focus principale è sull'animazione precisa, soprattutto in termini di accuratezza del tempo impiegato per percorrere la curva. La verifica della qualità viene fatta tramite calcoli manuali.
* Gestione dei bug: I bug che impediscono l'esecuzione del programma vengono risolti immediatamente. I bug meno critici vengono gestiti in un secondo momento e affrontati quando diventano necessari per il progresso del progetto.

**10. Pacchetti di lavoro (Workpackages)**

Il lavoro è suddiviso nei seguenti pacchetti principali:

* Project plan: Gestito da Nicolò, che si occupa della pianificazione e della gestione del progetto.
* Diagrammi UML: Creazione e mantenimento dei diagrammi UML (classi, stati, casi d'uso) affidato a Marco.
* Codice per la rappresentazione delle curve: Marco si occupa della scrittura del codice per la generazione e rappresentazione delle curve (compresa la circonferenza con raggio variabile).
* Implementazione dell'animazione: Marco è responsabile dell'implementazione e ottimizzazione dell'animazione della sfera lungo la traiettoria.
* Creazione dell'interfaccia: Il team si occupa della progettazione e implementazione dell'interfaccia utente per il simulatore.
* Documentazione: Nicolò si occupa di mantenere aggiornata la documentazione del progetto.

Attività principali: Le attività su cui il team ha investito più tempo sono la progettazione e i calcoli matematici per la corretta rappresentazione delle curve, in particolare la circonferenza con raggio variabile.

**11. Risorse**

* Risorse hardware e software:
  + Computer: Utilizzati per lo sviluppo e la progettazione del software.
  + Eclipse: IDE utilizzato per lo sviluppo del codice in Java.
  + Maven: Gestione delle dipendenze e del ciclo di build.
  + GitHub: Strumento di controllo di versione per la gestione delle modifiche e la collaborazione.
  + Papyrus: Utilizzato per creare i diagrammi UML (classi, stati, casi d'uso).
  + GeoGebra: Utilizzato per la verifica grafica delle curve e dei calcoli matematici.
* Risorse umane:
  + Marco Locatelli: Sviluppo del codice, creazione delle curve e implementazione dell'animazione.
  + Nicolò Nava: Gestione del project plan, documentazione e supporto nella progettazione.
  + Professore Gargantini: Revisore esterno del progetto, fornisce supporto e feedback.

**12. Budget e programma**

* Budget:  
  Non sono previsti costi aggiuntivi poiché si tratta di un progetto accademico.
* Programma:
  + Fase 1: Project Plan (scadenza dicembre)
    - Tempo stimato: 10 ore
    - Dettagli: Sviluppo completo del project plan, con documentazione delle risorse, milestone, rischi e pacchetti di lavoro.
  + Fase 2: Progettazione e rappresentazione delle curve (scadenza gennaio)
    - Tempo stimato: 15 ore
    - Dettagli: Sviluppo del codice per le curve e verifica matematica, con attenzione alla rappresentazione della circonferenza e altre curve preimpostate (parabola, retta, cicloide).
  + Fase 3: Implementazione dell'animazione
    - Tempo stimato: 15 ore
    - Dettagli: Codifica e ottimizzazione dell’animazione, assicurando che la sfera segua le curve senza problemi di prestazione.
  + Fase 4: Test e ottimizzazione finale
    - Tempo stimato: 10 ore
    - Dettagli: Test completi e ottimizzazione del simulatore, con eventuali modifiche basate sui risultati dei test.

Scadenze chiave:

* Project Plan completato: Dicembre
* Completamento applicazione e test finale: Gennaio