

**Mestrado em Engenharia Eletrotécnica e de Computadores**

Roller Coaster

Robótica Avançada

Grupo I

Ricardo Rodrigues nº11611

José Rodrigues nº10227

Rui Carvalho nº12634

Filipe Rodrigues nº12560

26 de Fevereiro de 2020

Resumo

O *resumo* do relatório (que só deve ser escrito após o texto principal do relatório estar completo) é uma representação abreviada e precisa, sem acrescento de interpretação ou crítica, escrita de forma impessoal, podendo ter, por exemplo, as seguintes três componentes:

1. um parágrafo inicial de introdução do contexto geral do trabalho.
2. resumo dos aspetos mais importantes do trabalho descrito no presente relatório, que por sua vez documenta o trabalho mais importante realizado durante o estágio. Deve mencionar tudo aquilo que foi feito, por isso deve concentrar-se no que é realmente importante e que deve ajudar o leitor a decidir se deve ou não consultar o restante relatório.
3. um parágrafo final com as conclusões do trabalho realizado.

**Palavras Chave (Tema):** Montanha russa, simulação, realidade virtual

**Palavras Chave (Tecnologias):** Software Unity, Visual Studio, Orange Edit, Oculus DK2

Índice

Índice de Figuras

[Figura 1 - Exemplo de imagens a) difícil leitura; b) fácil leitura 17](#_Toc313398608)

[Figura 2- Exemplo de lista de bibliografia 20](#_Toc313398609)

Índice de Tabelas

[Tabela 1 - Estilos pré-definidos 16](#_Toc313398617)

[Tabela 2 - Exemplo de tabela 18](#_Toc313398618)

Notação e Glossário

|  |  |
| --- | --- |
| **CAD** | Computer Aided Design |
|  |  |
|  |  |
|  |  |

# Introdução

Neste capítulo é apresentado um breve enquadramento do trabalho realizado, assim como os objetivos propostos para a realização do mesmo. Em simultâneo, são também descritas as tecnologias que foram necessárias para a sua realização, bem como os contributos académicos resultantes da execução deste trabalho.

## Enquadramento

## Tecnologias utilizadas

Para a realização deste trabalho foram indispensáveis alguns softwares nomeadamente o Unity IDE, OrangeEdit IDE, Qt Creator IDE, Visual Studio Community.

## Contributos deste trabalho

Este trabalho prático permitiu adquirir conhecimento de programação do robô Kuka, assim como o estabelecimento da comunicação de um motor de jogo com o próprio robô.

## Organização do relatório

Apresentação sucinta dos capítulos que fazem parte do relatório, descrevendo em poucos parágrafos o que cada um deles irá tratar.

# Comunicação aplicação - Kuka

Neste capítulo está representado todo o sistema desenvolvido para estabelecer um sistema de comunicação entre o nosso sistema de simulação e o robô KUKA.

O robô KUKA já tinha equipado plataforma de comunicação open-source “JOpenShowVar” logo era necessário desenvolver no nosso IDE método de comunicação com esta plataforma devido a que a “dll” disponibilizada pelo professor não ser compatível com o IDE utilizado. A “dll” foi desenvolvida com recursos a componentes da framework “Qt”, sendo esta incompatível com a que foi utilizada para a realização do projeto, Unity.

Para isso foi necessário desenvolver uma classe que foi denominada de “Client” pois é responsável por criar um cliente TCP e gerir toda a comunicação com o servidor.

Além das funcionalidades clássicas de classes para comunicação como o “connect”, “disconect”, esta tem de codificar e descodificar transmissões segundo a API (Application programming interface) que o servidor utiliza.

Para tal obedeceu-se à seguinte estrutura inicial quando o cliente inicia a transmissão. Após esta estrutura estar definida é acrescentado os bytes referentes à variável a ler/escrever e também o novo valor da variável caso seja uma operação de escrita.

Uma imagem com texto

Descrição gerada automaticamente

Foi decidido utilizar um sistema assíncrono de comunicação realizado por callback de forma a que o sistema não fique comprometido com as tarefas de comunicação.

## Aquisição de pontos do robô KuKa

Ainda na fase embrionária do projeto, tendo em consideração as limitações que poderiam existir no que se refere às restrições dos eixos do robô Kuka, optou-se pela aquisição de pontos em tempo real correspondentes a movimentos dentro de limites de controlo do Kuka.

Os pontos adquiridos foram conseguidos a partir de uma “thread” executada em “QT” que faz a aquisição de pontos com uma frequência de 1 segundo, sendo posteriormente armazenados num ficheiro com extensão “.CSV”. Seguidamente, no software *Unity* é carregado o ficheiro pela aplicação e consequentemente percorrido linha a linha, retirando os valores pretendidos para uma lista, onde é feito um tratamento de dados devido a particularidades da linguagem C#. No seguimento deste procedimento, é instanciado um vetor de 3 posições, que tem como finalidade armazenar as posições X, Y, Z para a criação do ponto no espaço.

## Criação da montanha russa

Após o ponto estar criado no vetor “pointPosition” é invocado o método “InsertNewPointAt2” da classe “spline” para inserir cada ponto na *spline* sequencialmente. Posteriormente é chamado o método “AutoConstructSpline()” que faz a fusão de todos os pontos usados no trilho da montanha russa.

Finalmente é requisitado o método “BuildRollercoasterTrack” responsável por gerar a montanha

## Geração de pontos no software Unity

# Conclusões

Bibliografia

[1] Autor 1, Autor 2 e Autor 3, (ano). Título, Editor.

[2] Autor (caso exista), título (caso exista), www.pagina\_internet.pt, <consultado a 12-12-2011>