# NLP & Games

Dinis Marques Firmino
P13240786

# Mini Project Idea

- Focused on natural language processing techniques and games.
- The ultimate means of interaction with game like environments.
- Little research and implementation of these techniques in games.
- Complexity of full NLP integration and compute power required largely increases depending on genre and scale of game.

**Benefits:**

- A different kind of immersive experiences
- Intuitive interaction
- Increased accessibility for people with disabilities involving motor movement

# Use cases within games

There are a range of potential use cases (some already exist) of NLP in interactive environments such as:

- Text adventure games
- Interactive educational games + gamification
- Open world games such as RPGs offer a huge potential for NLP and AI.
- RTS games
- Allowing natural language communication between game agents.
- Making single player games less single.

# System Design Overview

- Simple graphics engine for rendering in OpenGL and placement of items in a scene.
- Hybrid system design.
- C++ core with embedded Python
- Python NLP module
- Speech recognition
- XML data driven scene loading and object properties.

# Speech Recognition

- Speech recognition was the first stage of the system development.
- Researched into various open source frameworks, i.e CMUSphinx
- Ended up using Microsoft SAPI 5.4 as it comes pre bundled with Windows 10.
- Nowhere near as accurate out of the box as Google Speech API, but allows for further voice training and adding specific words into dictionary which made it feasible to be used in this demo project.
- Commercial APIs such as Google Speech API were by far the most accurate in transcribing speech even in noisy environments.
- Required extra work on application to establish client-server communications.

# Natural Language Processing Module

Python script using the NLTK framework for NLP + Boost.Python for embedding into C++.

A sentence and a response object is passed to the module.

When this script runs it performs:

- Tokenization
- Part-Of-Speech tagging (Creating tuples of each words and its grammatical tag)
- Chunking (Grouping by particular sequences of tags)
- Named entity recognition
- Retrieves characteristics of entities i.e adjectives
- Handles simple and specific command statements

# Mapping commands to game behaviour

- Where NLP intersects with HCI and Game AI
- Speech commands have to somehow be mapped to a specific agent or engine behaviour, i.e knowing which agent state to trigger, referred entity...
- Commands have to be related to the game world in context.
- In order to meet constraints of available agent/game states.
- The **CommandInterpreter** handles retrieved information from the NLP module.
- The state switching is determined by a simple finite state machine.

# Demo capabilities

Simple 1 sentence commands.

Features only 1 game agent called Bob.

Features 3 possible locations, England, France, Germany

Enough to demonstrate basic state switching using speech commands.

Can handle moving agents to a location.

Currently only supports moving objects, however more states can be added easily.

# DEMO

# Future Improvements

Necessary improvements:

- Incorporate other game AI techniques such as pathfinding into states.
- Improve FSM to support goal driven states
- Use a more robust game engine.
- Train the ASR and NLP with corpus of in-game items, names, etc…
- Handling a larger variety of commands and more complicated ones.

Major improvements:

- At the moment the application doesn’t feature real learning or A.I!
- Agents could personalities and full context awareness which affect interactions.
- Persistent knowledge between conversations.
- Complex dialogs with agents which could replace dialog choices.

# Q&A