# Aprendizado de Python com Realidade Aumentada

**Alunas: Clara Yuki Sano e Júlia Melo Teixeira dos Santos**
**Orientador: Prof. Dr. Carlos Hitoshi Morimoto**

Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo

## Motivação e objetivo

O ensino introdutório de computação às crianças e aos jovens é fundamental para que seu papel diante dos recursos digitais não seja apenas de consumo, mas também de produção e reflexão crítica. Apesar das novas gerações apresentarem fluência digital com naturalidade, o aprendizado inclusivo e formal da lógica de programação é indispensável para o cultivo de autonomia e incentivo de contribuição para comunidades digitais [1]. Diante desse contexto, este trabalho propõe e executa a criação do **PyBlox**, um jogo que promove o aprendizado da lógica de programação em Python através da imersividade dos recursos de Realidade Aumentada.

Sua conceptualização foi fortemente inspirada em **Scratch**, uma linguagem de programação visual baseada em blocos criada em 2007 pelo MIT. Sua interface virtual permite que os usuários interajam à moda *drag and drop* com trechos de código para que sua combinação resulte em um código completo e funcional. Tal mecanismo interativo é fundamentado na percepção de que jovens são pensadores que constroem conhecimento ao engajar ativamente em experiências de aprendizagem [2].

As qualidades práticas e imersivas do Scratch são extrapoladas no PyBlox com dois principais adicionais:

1. **Realidade Aumentada (AR)**: permite enraizar a interação com o software no mundo físico, através da manipulação dos blocos, e complementá-la pela interface virtual, através da projeção dos trechos de código sobre os blocos e da simulação do código final na tela de dispositivos Android.
2. **Python**: permite que o aprendizado obtido através do jogo seja diretamente aplicável em tarefas reais por ser uma das linguagens de programação mais populares da atualidade e amplamente utilizada nos meios profissional e acadêmico.

## Tecnologias

A implementação do PyBlox priorizou tecnologias que favorecessem a prototipagem ágil e a abstração de complexidade de hardware. A *game engine* **Unity** foi adotada como principal ferramenta de desenvolvimento devido a sua compatibilidade com múltiplas plataformas e seu caráter acessível. Sua *framework* **AR Foundation** e a biblioteca **ARCore** da Google agiram como complementos especializados em funcionalidades de AR, como percepção do ambiente real e rastreamento de blocos. Por fim, a plataforma de serviços de computação em nuvem **Google Cloud** foi empregada como núcleo adicional de computação responsável pelo processamento de algoritmos de visão computacional da *framework* **OpenCV** e pela execução remota de código em Python gerado a partir do arranjo dos blocos.

## Execução de código em Python

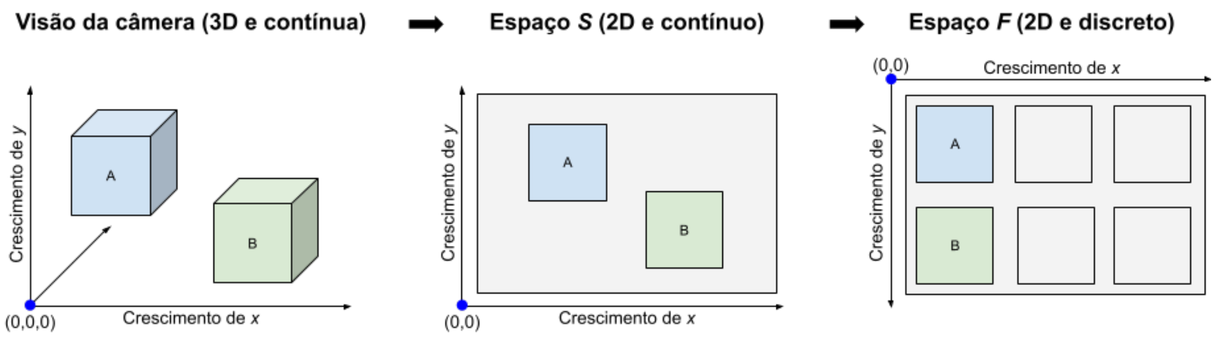
O PyBlox possui o requisito fundamental de executar o código em Python criado pelo usuário. Por ser uma aplicação desenvolvida em Unity (que trabalha nativamente com **C#**) e voltada para aparelhos **Android**, pensar em uma forma de executar código em Python foi um desafio grande no desenvolvimento.

A solução adotada foi delegar a execução do Python para um **servidor remoto**. Foi configurada uma função *serverless* no **Google Cloud** que recebe requisições HTTP contendo o código Python em formato JSON, executa o código em um ambiente isolado e retorna a saída ou erros também em JSON. No lado da Unity, a comunicação foi implementada usando `UnityWebRequest` [3] para enviar requisições POST assíncronas e `UniTask` para gerenciar o fluxo com `async/await`.

## Geração de código a partir do arranjo de blocos

A geração de código é fundamentada no rastreamento contínuo dos blocos físicos pelo componente `ARTrackedImageManager`. Suas coordenadas tridimensionais (3D) no mundo real são detectadas e mapeadas para suas coordenadas bidimensionais (2D) na tela do dispositivo ($S$) através do método `WorldToScreenPoint` da câmera. Para que o arranjo dos blocos seja logicamente interpretado como um arquivo de código ($F$), estabelece-se uma lógica de ordenação que simula as convenções de leitura:

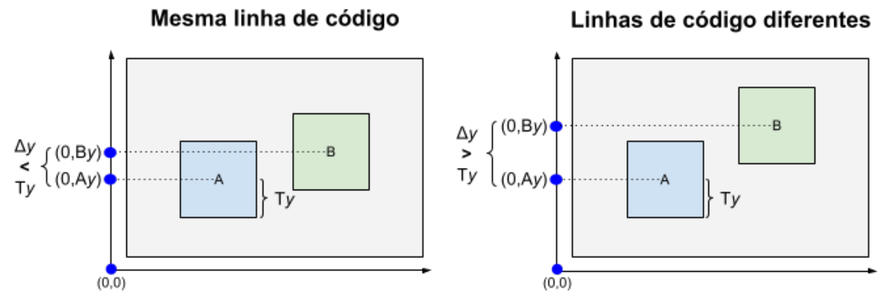
- Comparação primária pelo eixo **vertical ($y$)** de forma **decrescente**, equivalente a ler linhas diferentes de cima para baixo.
- Comparação secundária pelo eixo **horizontal ($x$)** de forma **crescente** caso a distância vertical dos blocos seja menor que a **tolerância** $T_y$, equivalente a ler a mesma linha da esquerda para direita.




## Detecção de bordas de blocos com visão computacional

A **tolerância vertical** $T_y$ é calculada a partir da **metade da altura** das imagens identificadoras de blocos. Para suprir a ausência de mecanismos nativos de detecção de dimensões de imagens no espaço da tela $S$, foi desenvolvida uma função em Python no **Google Cloud Run** responsável por detectar as bordas de blocos utilizando a *framework* OpenCV. Seu funcionamento consiste em processar um *frame* da câmera aplicando **filtro Gaussiano**, algoritmo de **detecção de bordas de Canny** e algoritmo de **aproximação poligonal de Douglas-Peucker** [4].

Os resultados são enviados à Unity, onde é realizado o pareamento entre os contornos dos quadriláteros e os blocos rastreados para, enfim, definir o valor dinâmico de $T_y$.




## Simulação de código executado

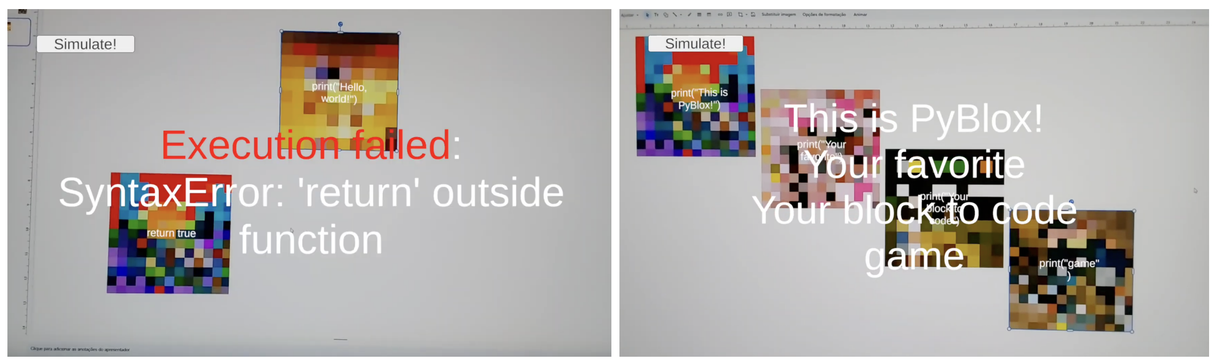
A versão final do PyBlox executa código Python que gera valores numéricos na saída padrão e os visualiza em uma *grid* 2D, permitindo que o usuário observe conceitos como progressões e sequências matemáticas de forma concreta.

A arquitetura usa eventos da Unity para desacoplar a execução da visualização. O módulo `PythonExecutor` dispara eventos ao iniciar a execução, ao receber sucesso ou ao encontrar erros. O `PythonSimulationController` escuta o evento de sucesso e processa a saída em duas etapas: primeiro, o método `ParseOutput` divide a saída em linhas e converte cada uma para valores numéricos válidos. Em seguida, `CreateSimulationPoints` mapeia esses valores para coordenadas 2D na tela.

O mapeamento considera a posição sequencial de cada valor no eixo $x$ e sua magnitude normalizada no eixo $y$, aplicando um *padding* de 0.1 nas bordas do *canvas* para melhor visualização. Os pontos são instanciados sequencialmente com um pequeno atraso entre cada um, criando uma **animação** que permite acompanhar visualmente o resultado do código construído.

## Resultados obtidos

Na versão inicial do PyBlox, o resultado da execução do código em Python montado pelo usuário era renderizado diretamente na tela do dispositivo, sobrepondo a visualização da câmera. As imagens abaixo demonstram exemplos da interface em cenários de falha e sucesso de execução, respectivamente:




Já na versão final, a interface é segmentada de modo a mostrar na parte superior direita da tela o código que será executado, e na parte inferior direita a simulação do código gerado (ou possíveis falhas de execução).


## Informações

Para mais informações, visite a página do projeto e o repositório com o código desenvolvido:

- https://tcc-mac0499.github.io/TCC-page/
- https://github.com/TCC-MAC0499/PyBlox

Ou escaneie o ***QR Code*** ao lado.

## Referências

Luciana Alvarez, **"Ensino de programação é aposta de colégios em todo o mundo"**, Revista Educação, 2014.

Annette lamb e Larry Johnson, **"Scratch: Computer Programming for 21st Century Learners"**, Teacher Librarian, v. 38, n. 4, 2011.

Unity Official Documentation, **"UnityWebRequest"**, 2015.

Yasoob Khalid, **"A guide to finding books in images using Python and OpenCV"**, 2015.