RoboCode

JOÃO PINTO - up201008649
Faculdade de Ciências da Universidade do Porto - DCC
and
JOSÉ SOUSA - up200803632
Faculdade de Ciências da Universidade do Porto - DCC
and
JULIANA FORTES - up200901637
Faculdade de Ciências da Universidade do Porto - DCC

Pensamento computacional é uma habilidade que todos deveriam adquirir. Quando dizemos todos, estamos naturalmente a incluir as crianças.
Se introduzirmos o pensamento computacional em crianças, ensinando-as a programar, vamos com certeza desenvolver o raciocínio destas, tornando-as adultos com características capazes de solucionar problemas de forma mais fácil e rápida.[1]

> ”Steve Jobs disse que toda a gente deveria aprender como se programa porque assim aprenderia a pensar.”

Categories and Subject Descriptors: [**Sistemas Multimédia**]: Projecto Final - Videojogos

Additional Key Words and Phrases: Education, kids, programming, game

1. INTRODUÇÃO

Em fevereiro, nomes gigantes da tecnologia como o fundador da Microsoft (Bill Gates), do Facebook (Mark Zuckerberg), do Twitter (Jack Dorsey), entre outros, juntaram-se a uma campanha lançada pela code.org para estimular o ensino de programação nas escolas. O projeto diz que nesta década serão criados mais de 1,4 milhões de empregos em programação, mas até 2020 os Estados Unidos formarão apenas 400 mil estudantes na área. Como a programação ainda não faz parte dos programas escolares do primeiro ciclo na maior parte das escolas no mundo inteiro, existem alguns aplicativos e ferramentas disponíveis na internet que tentam preencher essa lacuna.
Este artigo consiste na apresentação de uma proposta de um projecto que assenta no desenvolvimento de um jogo 2D, cujo objectivo é introduzir o pensamento algoritmo e técnicas de programação.
Numa primeira fase, o jogo foi pensado para aplicar em crianças, mas como se trata de um jogo de introdução à programação, as restantes faixas etárias não estão excluídas.
De uma maneira divertida, o utilizador, através de uma linguagem de comandos, poderá controlar os movimentos de um robot ajudando-o a cumprir missões a cada nível. Desta forma, este jogo pretende estimular o interesse pela programação. Com o RoboCode pretende-se construir um jogo capaz de estimular o cérebro de uma criança. Escolhemos as crianças como público alvo, porque podemos juntar a vertente de arendizagem escolar, como a matemática, ao pensamento algoritmico.

2. ESTADO DA ARTE

Para este jogo, foi um realizado um estudo para saber o que se encontrava disponível nesta área actualmente. O site *Read Write Web* separou seis plataformas, voltadas para crianças com idades compreendidas entre os 5 e os 15 anos, que queiram aprender a programar. [2]

O **Codeacademy**, recomendado a partir dos 12 anos á uma plataforma que pretende ensinar programação a qualquer pessoa. A sua interface não é tão atraente e colorida como podemos encontrar em outros sites contudo, é amigável e simples de entender. Pode ser aprendido as linguagens: Python, Ruby, PHP, HTML e JavaScript.
Esta plataforma defende uma campanha que consiste em que os jovens treinem programação depois das aulas, incentivando à criação de clubes de programação em escolas.

Fig. 1. Codecademy

O **Scratch**, recomendado a partir dos 8 anos, é uma plataforma criado pelo MIT (Massachusetts Institute of Technology) com parceria da Microsoft, Intel e Nokia.
Usa uma linguagem de programação que facilita a criação de histórias interativas, animações, jogos, música e arte.
Segundo os criadores, à medida que os jovens criam e compartilham projetos na ferramenta, aprendem importantes ideiais matemáticas e computacionais, ao mesmo tempo em que aprendem a pensar criativamente.

Fig. 2. Scratch

O **Code Monster**, recomendado dos 9 aos 14 anos é uma app que contém um monstro que orienta crianças sobre como alterar variáveis do Javascript. Ao fazerem estas alterações, modificam a aparência dos blocos numa tela ao lado. A cada nova lição traz novas cores e formas para os blocos.

Fig. 3. Code Monster

O **Alice**, recomendado a partir dos 8 anos, foi criado por pesquisadores da Universidade de Virgínia como uma introdução à programação orientada a objetos.
Nesta plataforma, os usuários aprendem conceitos fundamentais de programação no contexto de criação de filmes animados e jogos de vídeo simples.
Os objetos 3D povoam um mundo virtual em que os usuários arrastam e soltam peças para criar o programa. Assim, eles conseguem ver a relação que existem entre as instruções de programação com o comportamento dos objetos na animação.

Fig. 4. Alice

O **Daisy the Dinosaur**, recomendado entre os 5 e os 8 anos, é uma app para iPad que pretende ensinar os conceitos básicos da lógica de programação para as crianças.
O funcionamento é simples: blocos com comandos como ”rolar”, ”saltar” e ”crescer” são arrastados para uma área de programação. Ao começar a app, os jogadores veem uma relação direta entre os comandos e as ações que o dinossauro executa.

Fig. 5. Daisy the Dinosaur

O **Hackety-Hack**, recomendado acima dos 13 anos é uma plataforma que apresenta uma introdução à linguagem Ruby. O programa tem uma interface dividida em duas telas: um editor para a introdução de comandos e testar programas e uma lição que esclarece o código. Depois do tutorial, os utilizadores criam e compartilham programas e jogos simples.

Fig. 6. Hackety-Hack

Além deste 6 sites já apresentados temos disponível mais um, o **code.org** que é um projeto que tem como objetivo introduzir as pessoas, de qualquer faixa etária, no mundo da programação. Os desenvolvedores deste projecto acreditam que a informática deve fazer parte do currículo, na educação, junto a outras disciplinas já ensinadas hoje como matemática, física, português.
Nesta plataforma que desenvolveram o utilizador aprende a programar em blocos e de maneira simples. Basta arrastar o bloco com a ação que o personagem deve fazer e mandar executar o programa. Existem vários modos e níveis que vão ficando cada vez mais complicados à medida que o utilizador evolui.

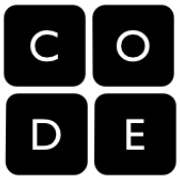

Fig. 7. code.org

Como podemos ver, já existe algum trabalho nesta área de ensino de programação a crianças, adolescentes e adultos sem formação em programação, e tende a ter um crescimento no decorrer dos próximos tempos.

3. DESIGN DO JOGO

O design do jogo apresentado neste artigo, RoboCode, foi pensado e construído como um jogo 2D. A escolha de um jogo 2D prende-se no facto de que este apresenta uma jogabilidade simples e a sua implementacão é de menor dificuldade em relação o a um jogo 3D, uma vez que o jogador apenas movimenta a personagem em dois eixos.

3.1 Personagem

A personagem principal do jogo é um robot (ver Fig. 8.).
A escolha de um robot para personagem principal do jogo foi natural pois um robot é, na sua essência, uma máquina que executa código, o que o torna adequado para o objectivo do jogo.
Esta personagem pode, de certo modo, estreitar o pensamento do jogador ao associar as habilidades e o mindset que ganha ao longo do jogo. Estas capacidades aplicam-se a vários outros problemas e situações, mas isso fica como uma ”surpresa” para o jogador, quando se deparar com novos problemas e descobrir que usou técnicas ou capacidades adquiridas neste jogo para os resolver.

Fig. 8. Robot

3.2 Cenário

O jogo ocorre num edifício. Este edifício é constituído por um conjunto de salas e cada uma das salas representa um nível do jogo.
Estas são constituídas por uma porta que dá acesso à sala seguinte. Para abrir a porta da sala, a personagem terá que coleccionar todos os cartões de acesso (ver Fig. 9.) espalhados por esta.

Fig. 9. Cartão de acesso

As salas (ver Fig. 10.), são formadas por tiles (ver Fig. 11.) de um tileset que, repetidos e devidamente posicionados formam as paredes, porta, e outros obstáculos.

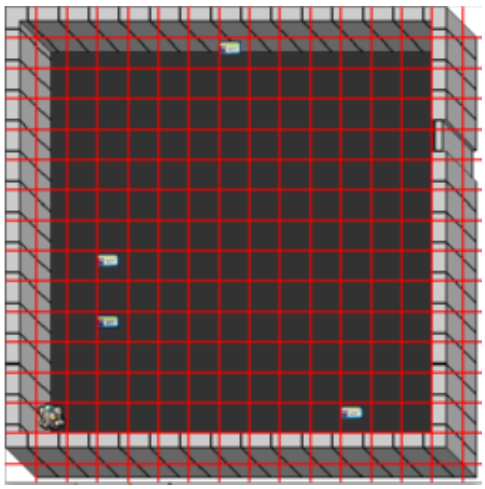

Fig. 10. Sala

Para programar o comportamento do robot, o jogador tem à sua disposição uma interface onde do lado direito pode visualizar a sala do nível onde se encontra e do lado esquerdo pode encontrar um editor de texto, onde colocará os comandos para controlar a personagem (ver Fig. 12.).

A sala do nível onde o jogador se encontra e que pode visualizar do lado direito, é auxiliada com a sobreposição de uma grelha para que o jogador mais facilmente calcule quantas unidades a personagem terá que se deslocar.
No editor de texto, após introduzir os comandos, o jogador pode

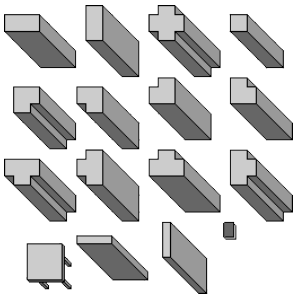

Fig. 11. Tileset

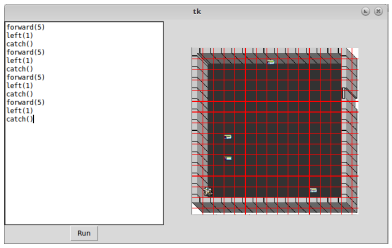

Fig. 12. Interface

testar o seu código, ou seja, o comportamento da personagem, clicando no botão que a interface disponibiliza.

Finalmente, quando o jogador chega ao final de cada nível é apresentada uma mensagem “Level Completed” (ver Fig. 13.)

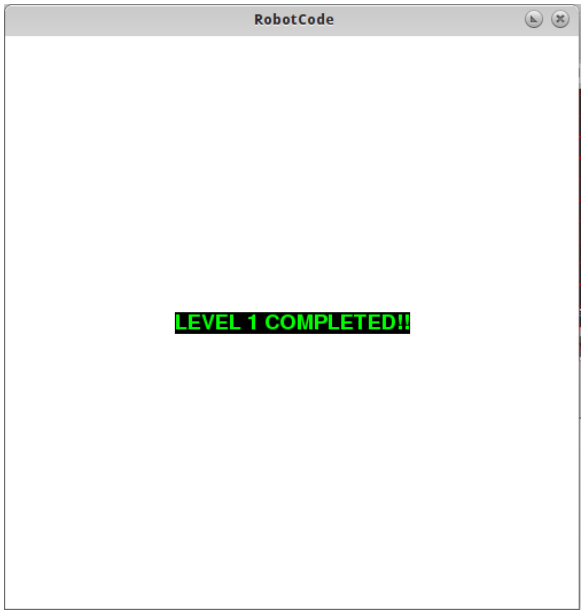

Fig. 13. Nível Completo

3.3 Objectivo do jogo

A personagem encontra-se num edifício com 3 andares, ou seja, 3 salas. O objectivo é abandonar o edifício. Para isso, terá de coleccionar os cartões de acesso espalhados na sala para abrir a porta e passar para a sala seguinte.

4. DESENVOLVIMENTO DO JOGO

O RoboCode foi desenvolvido usando a linguagem Python e os seus módulos pygame, tkinter e ply.
Para o editor de código foi usado o tkinter, pela simplicidade da interface pretendida (apenas uma janela com uma textbox, uma imagem e um botão).
No simulador do robot foi usado o pygame, por ser um módulo vocacionado para o desenho 2D e cuja gestão de eventos se mostrou bastante compatível com o que se pretendia. Essa gestão de eventos permitiu criar um compilador que compila a linguagem de comandos do robot em código Python que lança os eventos programados. O código é passado ao simulador como o "programa" que o robot vai executar, sendo efectivamente executado no momento de inicialização do mundo (sala) do nível corrente.
Para o compilador usamos o módulo ply do python que nos permite definir um lexer e um parser de maneira simples.
Esta estrutura modular do jogo permite o posterior desenvolvimento, em separado, dos vários módulos:

—O IDE pode receber sintax highlighting;

—O compilador pode ser desenvolvido para adicinar complexidade à linguagem;

—O editor de níveis onde se pode adicionar,por exemplo, o parsing de novos tipos de objectos disponíveis para criar os níveis;

—O simulador pode (apenas comentando meia linha de código) ser adaptado para ser jogado separadamente e controlado com o teclado, bem como novas features podem ser adicionadas, como novas ações para o robot. . . .

4.1 Python

Python é uma linguagem de programação de alto nível, interpretada, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991. Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerido pela organização sem fins lucrativos Python Software Foundation. Apesar de várias partes da linguagem possuírem padrões e especificações formais, a linguagem como um todo não é formalmente especificada. O padrão de facto é a implementação CPython.
A linguagem foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código sobre a velocidade ou expressividade. Combina uma sintaxe concisa e clara com os recursos poderosos de sua biblioteca padrão e por módulos e frameworks desenvolvidos por terceiros.[3]

4.2 Pygame

Pygame é uma biblioteca de jogos multiplataforma, independente de sistema operativo, desenvolvida para ser utilizada em conjunto

Fig. 14.

com a linguagem de programação Python. O nome desta biblioteca teve origem da junção de Py, proveniente de Python e Game, que significa Jogo, ou seja, Jogos em Python.
Pygame baseia-se na ideia de que as tarefas mais intensivas a nível computacional num jogo podem ser abstraídas separadamente da lógica principal, ou seja, o uso de memória e CPU (úteis para processar imagens e sons) são tratados pelo próprio código do Pygame e não pelo código do seu jogo. Assim, torna-se possível utilizar uma linguagem de alto nível, como Perl ou Python para organizar a estrutura do jogo em si. [4]

Fig. 15.

4.3 Tkinter

Tkinter é a biblioteca padrão da linguagem Python ou seja, acompanha a distribuição oficial do interpretador Python.
Os programas escritos usando a Tkinter tem portabilidade entre Linux, Windows e Mac, além da garantia de que qualquer um poderá executar o programa sem precisar instalar bibliotecas extras. Tem uma API simples de se aprender e fácil de lembrar e é muito bem documentada, com inúmeros tutoriais e referências de ópima qualidade disponíveis na Web.
Como desvantagens temos o facto de faltar muitos componentes importantes, como notebooks e combo box (apesar de estes poderem ser feitos combinando outros componentes). Alguns elementos (listbox por exemplo) são incompletos ou limitados. Em compensação o componente Text é muito poderoso. A Tkinter usa uma função própria para aceder directamente às funções do sistema operativo e desenhar os seus próprios elementos no ecrã. Ou seja, tem a vantagem de ser mais facilmente portável, mas a grande desvantagem de ter uma aparência diferente da nativa do sistema operativo.[5]

4.4 Ply

O PLY é uma ferramenta de parsing escrita puramente em Python. Básicamente trata-se de uma reimplementação do Lex e do Yacc da linguagem C. Ao contrário so Lex e do Yacc em C, que usam a técnica de parsing LALR, o PLY usa a técnica LR que pode facilmente incorporar gramáticas de grande dimensão. O PLY também tem um modo de debugging exaustivo e contém ferramentas próprias que reportam os erros.[6]

4.5 Controlo do Jogo

Para controlar o robot, foi implementado uma linguagem de comandos que manipula a orientação, o movimento e as ações do robot. O ”programa” é compilado para um subconjunto de intruções de Python que é então passado ao simulador.

Guia de comandos:

forward(n) : Move o robot n unidades para a frente.

backward(n) : Move o robot n unidades para trás.

right(n) : Altera a orientação do robot n*90º para a direita.

left(n) : Altera a orientação do robot n*90º para a esquerda.

catch() : Recolhe o cartão de acesso (se existir).

Depois de inserir o conjunto de comandos é só clicar no botão ”Run” para que o robot execute as instruções escritas no editor de código.

5. RESULTADOS

O resultado deste projecto é um jogo com 3 níveis (3 salas) em que o jogador tem de construir um conjunto de comandos (escrever o seu programa) que o fará ultrapassar os diferentes desafios do nível onde se encontra e assim passar para o nível seguinte.
A cada nível, o jogador consegue interiorizar pequenas técnicas de pensamento algoritmico e o grau de dificuldade e a exigência vai aumentando. Atraves de boas mecânicas de jogo é possível tornar a acção de jogar num hábito. Através desse hábito será possível reforçar e introduzir comportamentos.
Para testar a veracidade do argumento acima mencionados, foram realizadas testes de usabilidade. Assim foram aplicadas experiências a duas crianças, sem qualquer tipo de conhecimento de programacão, e foi constatado que, após uma explicacão das regras e da linguagem, estes rapidamente desenvolviam algoritmos e compreendiam chamadas a funções bem como passagem de argumentos.
Os jogadores conseguem usar aquilo que aprendem no jogo em varias situações. Esse tipo de aprendizagem tem a designação de ”situated learning”.

6. CONCLUSÕES

Este jogo, ainda que numa primeira versão, tem uma linguagem um pouco limitada mas pode ser visto como uma plataforma com potencial para evoluir. Novas *features* podem ser facilmente adicionadas à linguagem, abrindo assim a porta para a transferência de novos conceitos de programacão ao jogador.
Com projectos como o RoboCode, passa ser possivel aprender um pouco sobre computação, mais especificamente aprender a ter pensamento algoritmico, de forma empolgante e divertida. Consequentemente, havendo mais projectos deste tipo, há mais possibilidades de escolha, o que pode proporcionar mais interesse por parte dos possiveis utilizadores.

7. REFERÊNCIAS

1. Filipe Fernandes. Porque toda criança deve aprender a programar. 2014.
2. TERRA NETWORKS S.A. Aplicativos ensinam programacão a crianças. 2013.
3. Python Software Foundation. python. 2014.
4. Python Software Foundation. pygame. 2014.
5. Tkinter — Python interface to Tcl/Tk. tkinter. 2014
6. PLY (Python Lex-Yacc). ply. 2011.