Justin B. Bird

CS 275

Hang Qi
Jian Gong
Lunbo Xu

Outline

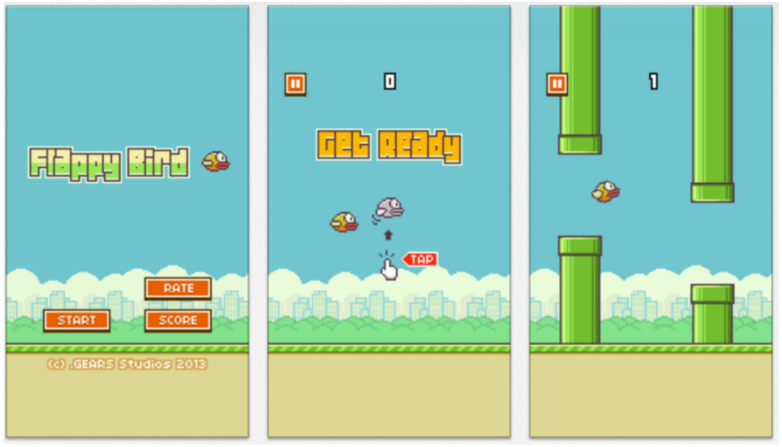
- Introduction
  - Background
  - Game design and tools used
- Modeling
- Learning
- Demo

Introduction

❑ Background

A recently popular mobile game app, on Android and IOS. Player controls a bird, attempting to fly between columns of green pipes without coming into contact with them.

Originally created by Nguyễn Hà Đông, a Vietnamese developer.



Then a plethora of Flappy Bird copies and clones emerged, such as Squishy Bird, and even MMO version...

Many clones...

Webkynang.vn

SQUISHY BIRD

Other: Kill others in your way ☺

Webkynang.vn

CLUMSY BIRD

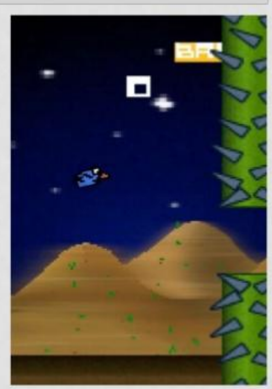
Other: Graphic and Phone orientation

Webkynang.vn

DESERT BIRD

Other:

Bring you the terrible feeling
with the TUBE like in the hell ☹

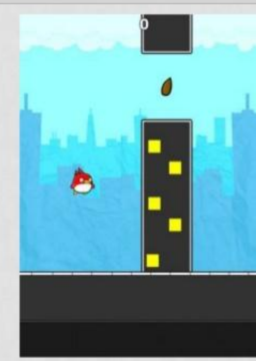


Webkynang.vn

CITY BIRD

Other:

Scenario:
Insiside City
and
Outside City



Our Game

- ❑ Play Mode

  How far can you fly the bird ?

- ❑ Learning Mode

  Behavior Model: reinforcement learning

Tools

- ❑ WebGL

  A built-in JavaScript API for rendering interactive 2D/3D graphics.

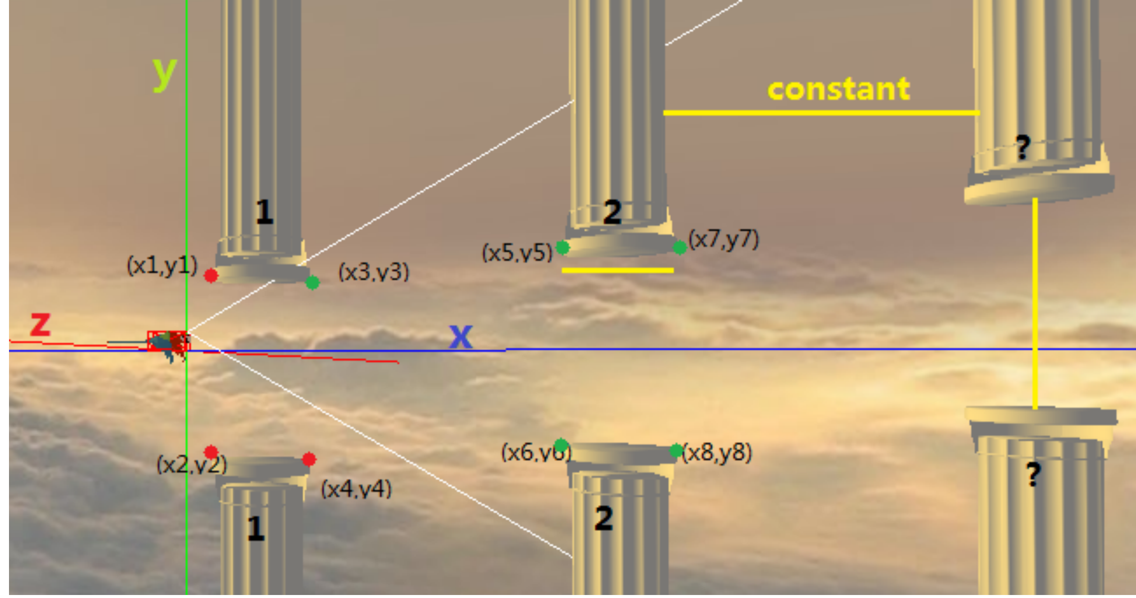
- ❑ Three.js

  A lightweight JavaScript framework for WebGL.

- ❑ Chrome

  JavaScript console and debugger.

Modeling



Real vision ? No

too big state space

Perception

2 pillars, 8 points, 16 float

known points: in the fovy && not occluded;
unknown points marked as -1

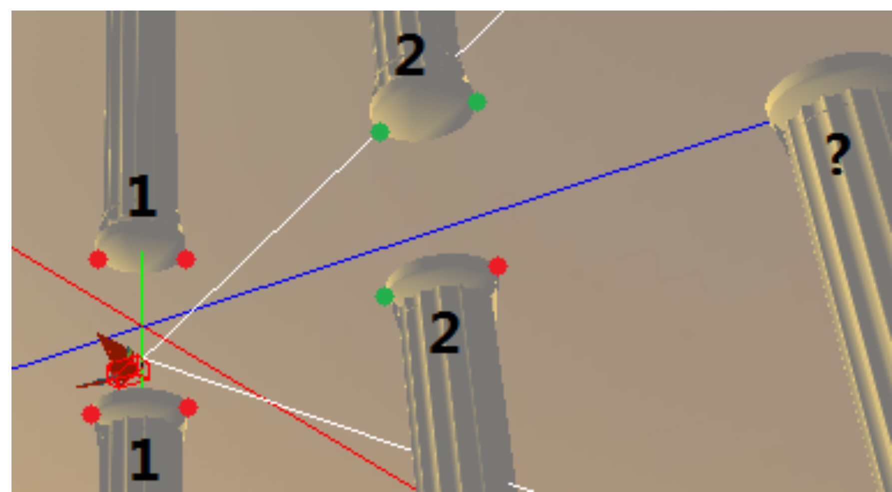
state space = {<x1, y1, x2, y2, …, x8, y8>}

Perception & short memory

know all 8 points

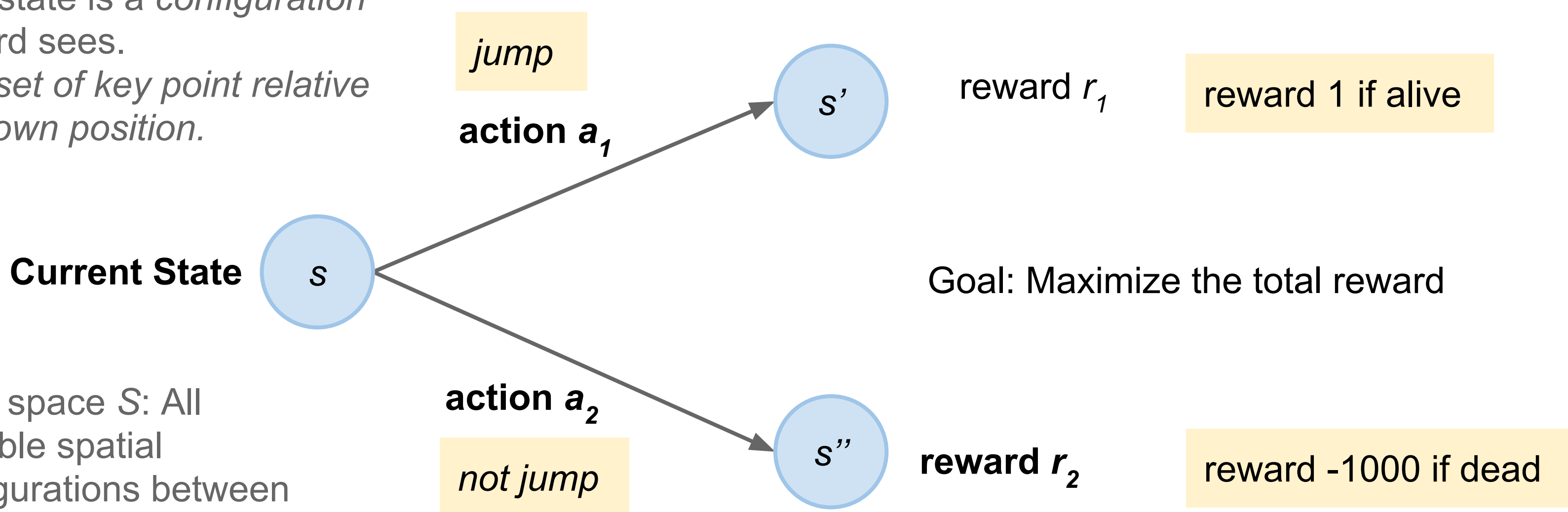
more clever

better results



Formulation: Markov Decision Process

Each state is a *configuration* our bird sees.
i.e. a set of key point relative to its own position.



Goal: Maximize the total reward

State space S: All possible spatial configurations between our bird and the key points.

We want to learn a ***decision policy*** $a_t = \pi(s_t)$ that maximize the total reward:

$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})$$

Q-Learning

If we know the exact *transition probability* and the *reward of the states*, we can solve the policy.

$$\pi(s) \ := \arg\max_a \Big\{ \sum_{s'} \underline{P_a(s,s')} \Big(\underline{R_a(s,s') + \gamma V(s')} \Big) \Big\}$$

Although in the game, the behavior of the bird is ***deterministic****. But as we didn't model the velocity into the state, the outcome of an action is* ***stochastic****.*

Using reinforcement learning, we want to learn the policy:

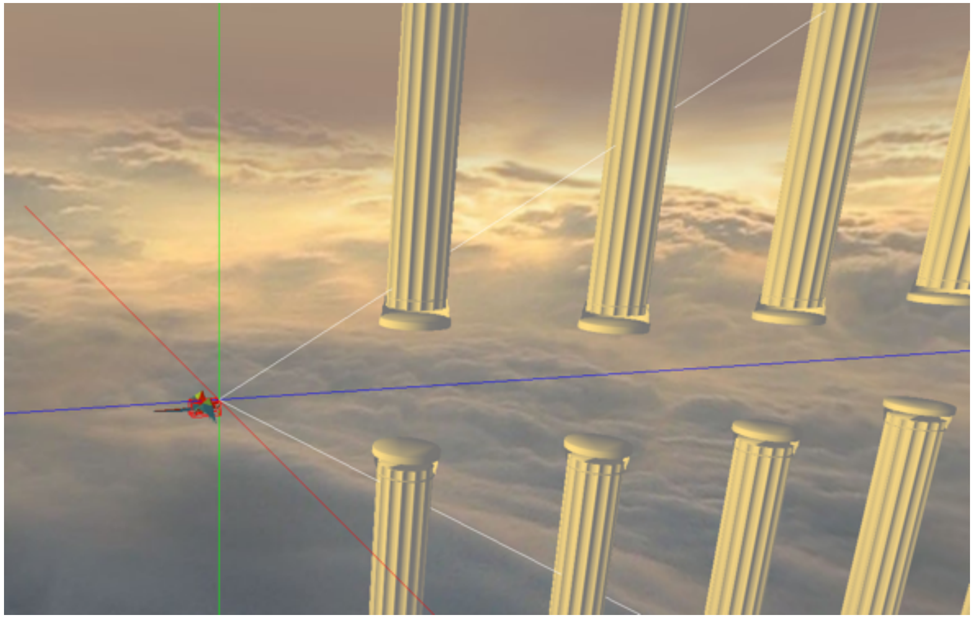
$$Q : \underset{\text{state}}{S} \times \underset{\text{action}}{A} \to \underset{\text{reward}}{\mathbb{R}}$$

Algorithm:

$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \times \left[\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t, a_t)}_{\text{old value}} \right]$$

Demo

- Perception & short-term memory
  - ~18K iterations to converge
- Perception only
  - smaller state space due to the occluded points
  - converge faster



Thank you!