Základy tvorby interaktívnych aplikácií

- Úvod do predmetu
- Ing. Jaroslav Erdelyi
- LS 2021-2022

Prednášajúci

- Ing. Jaroslav Erdelyi
- E-mail: jaroslav.erdelyi@stuba.sk
- Kancelária: 5.04
- Konzultačné hodiny: dohodnúť sa mailom

Cvičiaci

- Ing. Vladimír Kunštár – vedúci cvičení
- Mgr. Kostiantyn Rudenko
- Ing. Richard Marko, PhD.

Riešenie problémov

- Otázky týkajúce sa hodnotenia, podmienok absolvovania predmetu a skúšky riešiť s prednášajúcim
- Otázky spojené s cvičeniami riešte:
  - -so svojím cvičiacim
  - -s vedúcim cvičení
  - -s prednášajúcim

Informácie k predmetu

- Prednášky
  - Praktické a teoretické informácie
- Cvičenia
  - Konzultácie a riešenie projektu
  - Riešenie úloh
- Všetky materiály budú na dokumentovom serveri v AIS

Podmienky absolvovania

- Hodnotenie
  - Skúška: 30 b
  - Projekt: 70 b
- Podmienky absolvovania predmetu
  - Minimum na zisk zápočtu – 35b z projektu
  - Minimum zo skúšky – 7b
  - Podrobné podmienky – AIS dokument Podmienky absolvovania

Projekt na cvičeniach

- Odovzdanie projektu v poslednom týždni semestra
- Min. 35b z projektu
- Aktívna účasť na cvičeniach
- Odovzdanie a predvedenie výstupov v kontrolných bodoch

Projekt na cvičeniach

- Vytvoriť jednoduchú interaktívnu hru
- HTML5 Canvas a JavaScript
- Individuálna práca na projekte
- Nutnosť použiť objektovo-orientovaný prístup
- Ďalšie podrobnosti v rámci cvičení

Kontrolné body

- KB1: **06.03.2021** do 23:59
  - dokumentacia k navrhu hry: **10b** (v pdf)
- KB2: **20.03.2021** do 23:59
  - dokumentacia s pripravenymi podkladmi: **3b** (v pdf)
- KB3: **07.04.2021** do 23:59
  - projektove subory zakladnej implementacie (v zip)
  - predvedenie rozpracovanej hry na cvičeniach (08.04.2021): **7b**
- KB4: **08.05.2021** do 23:59
  - projektove subory finalnej implementacie, vratane grafickych a zvukových súborov (v zip)
  - jeden zaujimavy obrazok z hry
  - predvedenie finálnej hry na cvičeniach (9-13.05.2021): **50b**

Ako úspešne absolvovať predmet

- Aktívna práca na cvičeniach
- Pravidelná príprava
- Priebežná práca na projekte
- Nebáť sa pýtať ak niečomu nerozumiem
- Účasť na prednáškach
- Samo-štúdium

Osnova predmetu

- Úvod do JavaScriptu
- Objekty a objektovo orientovaný prístup
- Techniky v rámci objektovo orientovaného programovani
- Návrhové vzory
- Spracovanie udalostí
- Sieťové aplikácie
- Backend, Frontend webových aplikácií
- Interaktívne aplikácie
- Dizajn aplikácií

Cieľ predmetu

- Nadobudnúť základné teoretické znalosti
- Nadobudnúť základné praktické skúsenosti s programovaním interaktívnych aplikácií pomocou webových technológií
  - JavaScript, OOP, DOM, jQuery
- Navrhnúť a vytvoriť základné riešenie

Čo nie je cieľ predmetu

- Pokročilé programovanie v JavaScript-e
- Vytváranie komplexných herných mechaník
- Implementácia počítačových a mobilných aplikácií

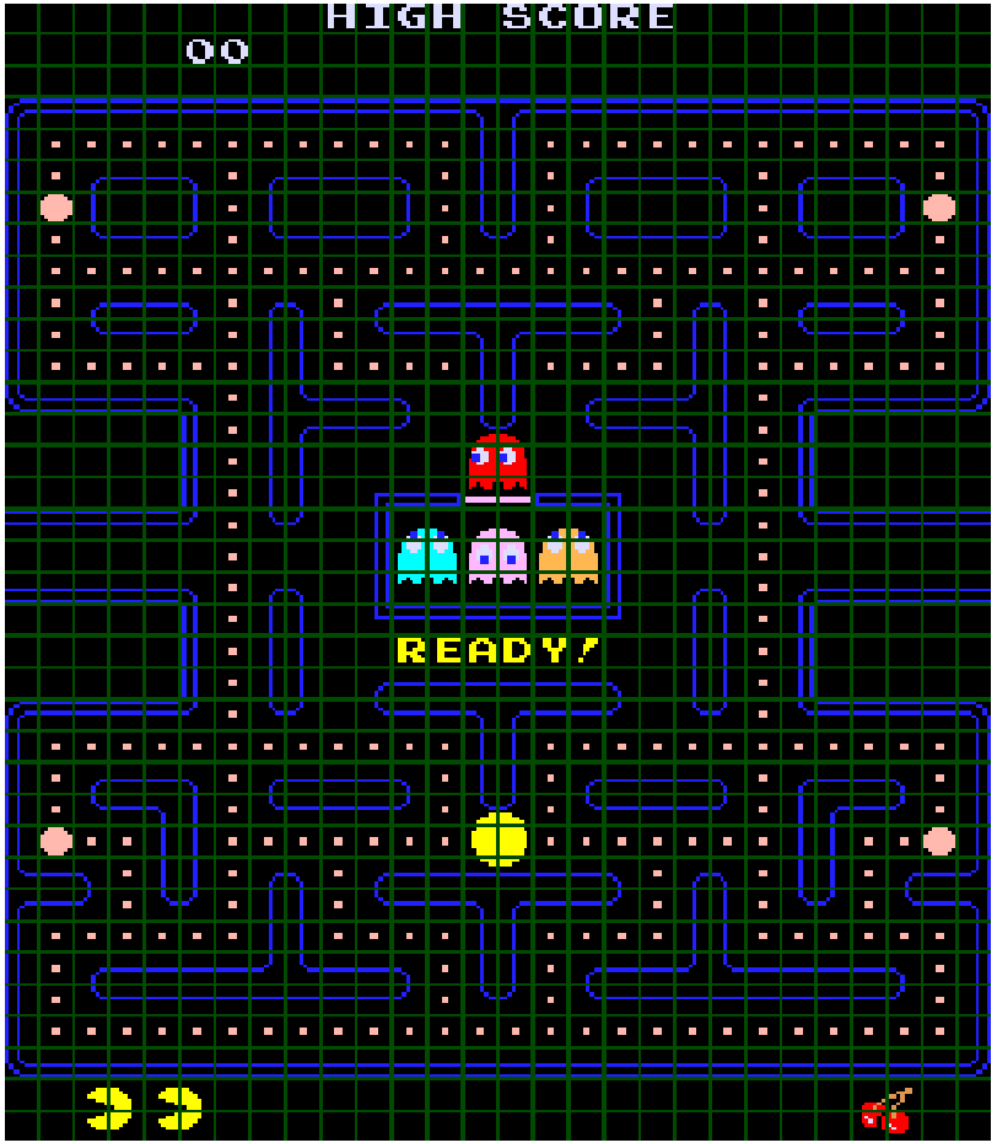
Otázky k predmetu

Základné herné dizajny

- Pac-Man
- Super Mario

Pac-Man

- Hracia plocha:
  - Bludisko
  - Žlté bodky
  - PowerUp
  - Duchovia
- Cieľ hry:
  - Pozbierať všetky žlté bodky

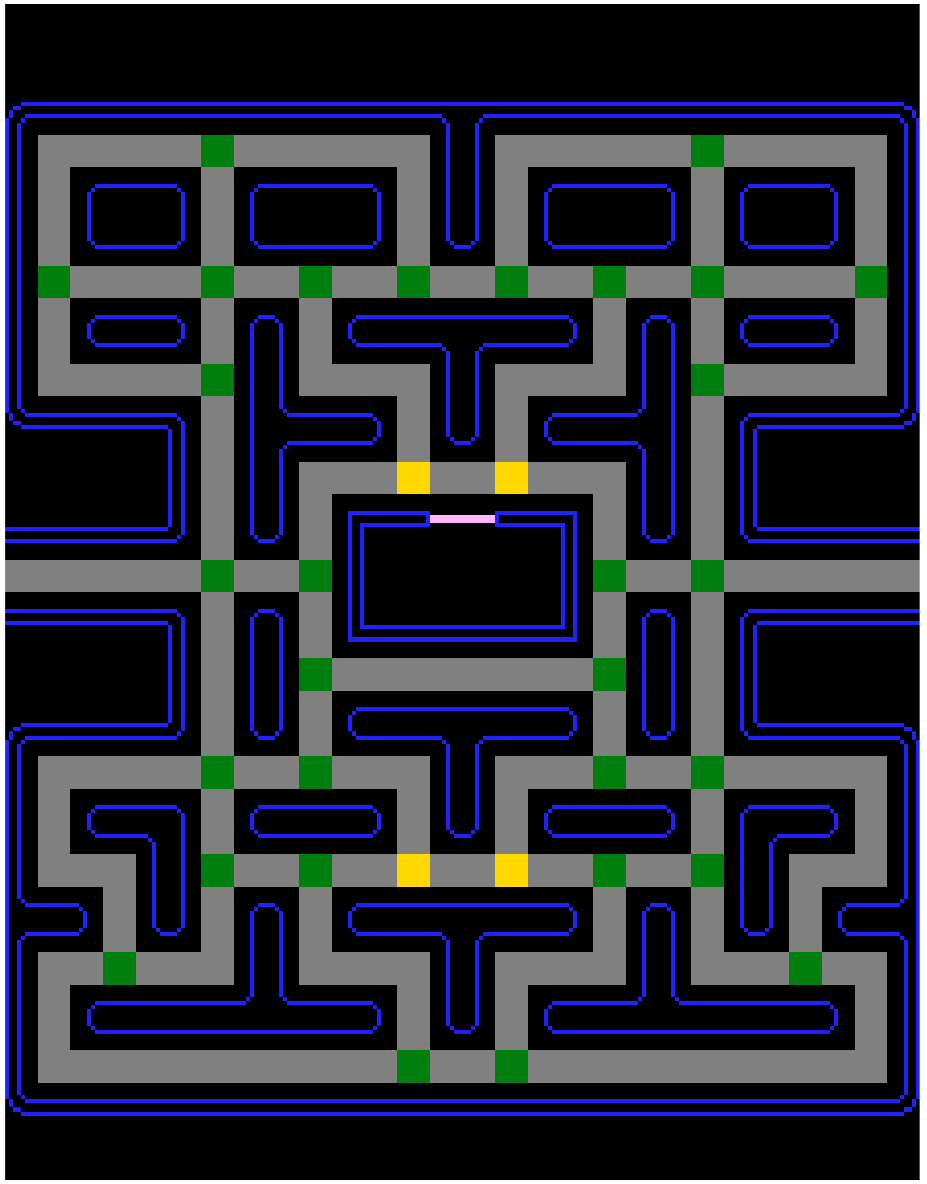


Duchovia

- Majú módy správania
  - Naháňanie, rozutekanie, vystrašenie
- Trvanie módov
  - 7s rozutekanie, 20s naháňanie
  - 7s rozutekanie, 20s naháňanie
  - 5s rozutekanie, 20s naháňanie
  - 5s rozutekanie, permanentné naháňanie

Duchovia

- Snažia sa dostať na určitú dlaždicu
  - Plánujú jeden krok dopredu
  - Rozhodujú sa na novej dlaždici – nesmie sa vrátiť
- Rozhodovanie duchov
  - Zelené
  - Žlté – v móde naháňania nesmú ísť hore

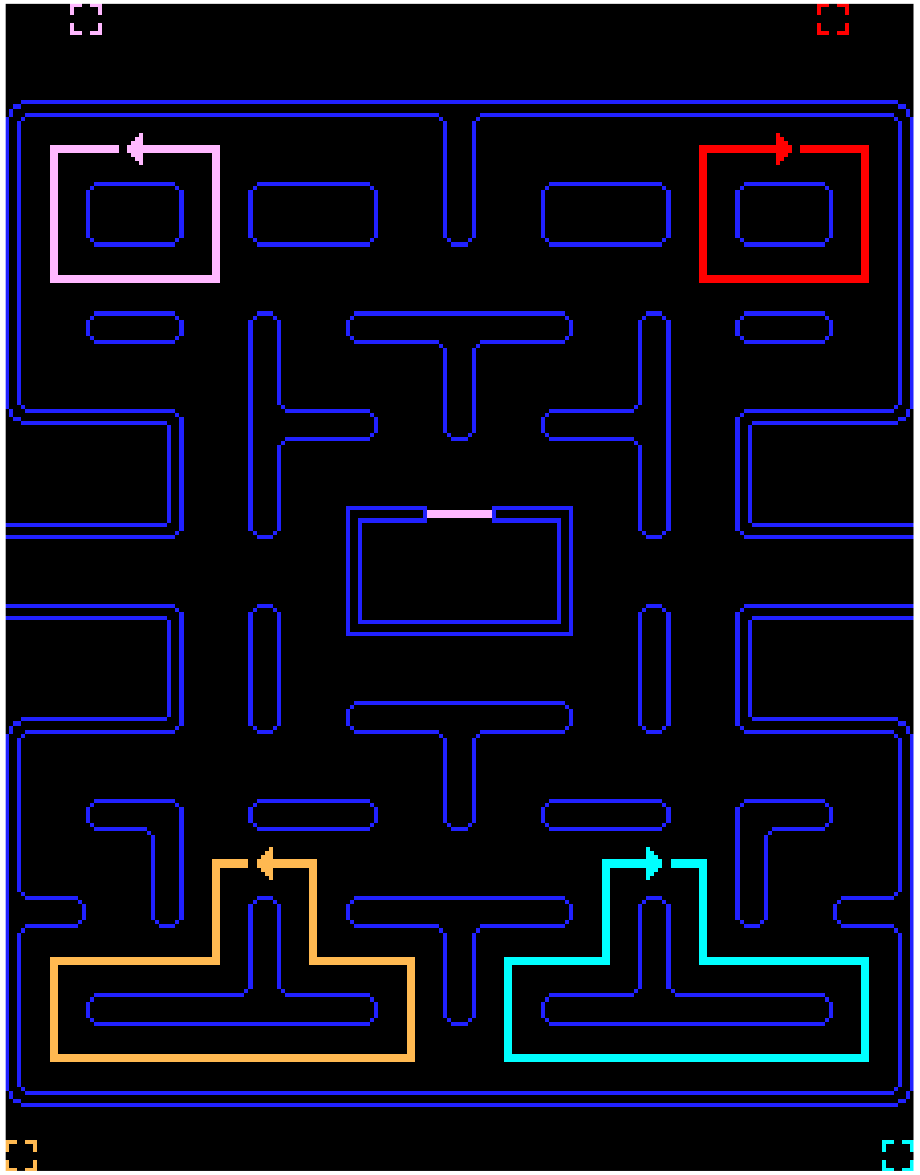


Duchovia

- Rozhodujú sa na základe vzdialenosti od cieľovej dlaždice:
  - Spravia aj zlé rozhodnutia
  - Ak je rovnaká vzdialenosť tak sa ide podľa priority – UP > LEFT > DOWN
- Duchovia majú mená aj definované rozhodovanie:
  - Červený
  - Ružový
  - Modrý
  - Oranžový

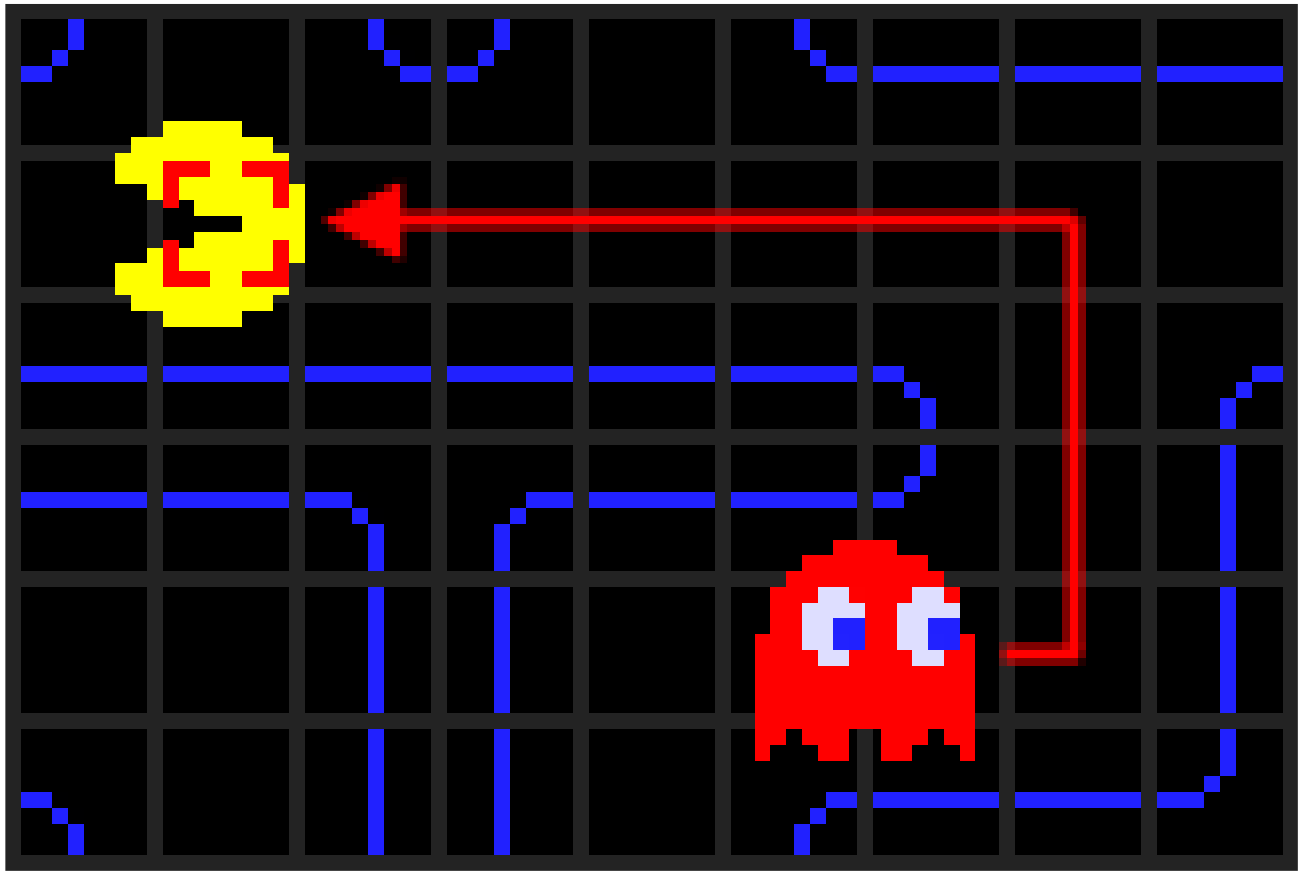
Mód rozutekania

- Duchovia majú svoju domovskú pozíciu
- V tomto móde sa tam snažia dostať



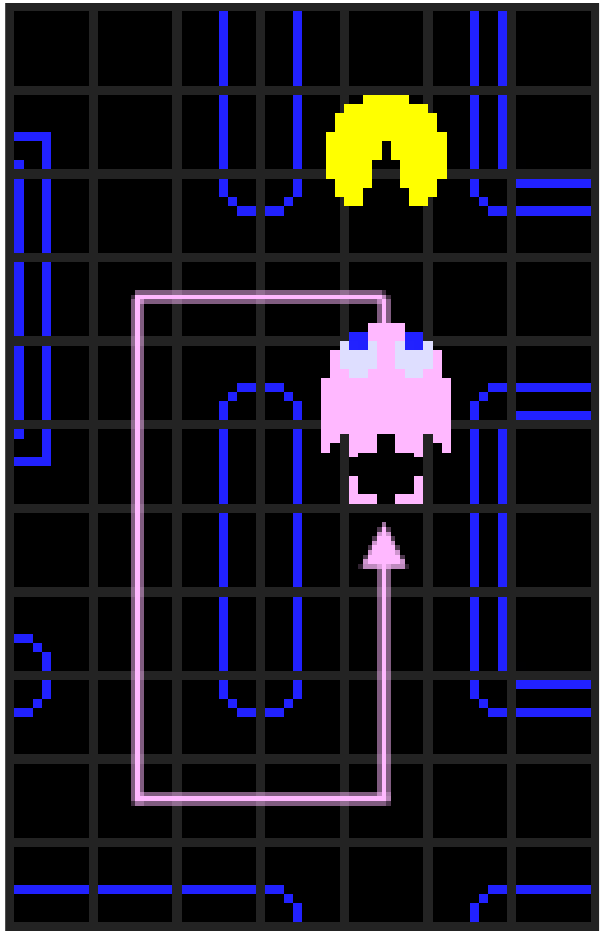
Červený duch - Blinky

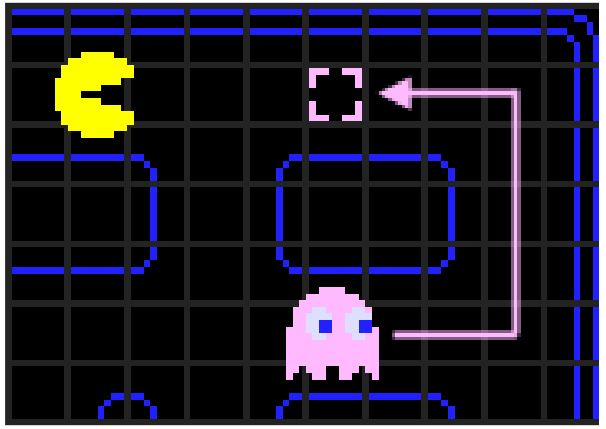
- Naháňa hráča – cieľová pozícia je aktuálna pozícia hráča
- Ide najkratšou cestou
- Rýchlosť pohybu sa zvýši o 5% podľa situácie v hre – čas, počet zostávajúcich žltých guličiek



Ružový duch - Pinky

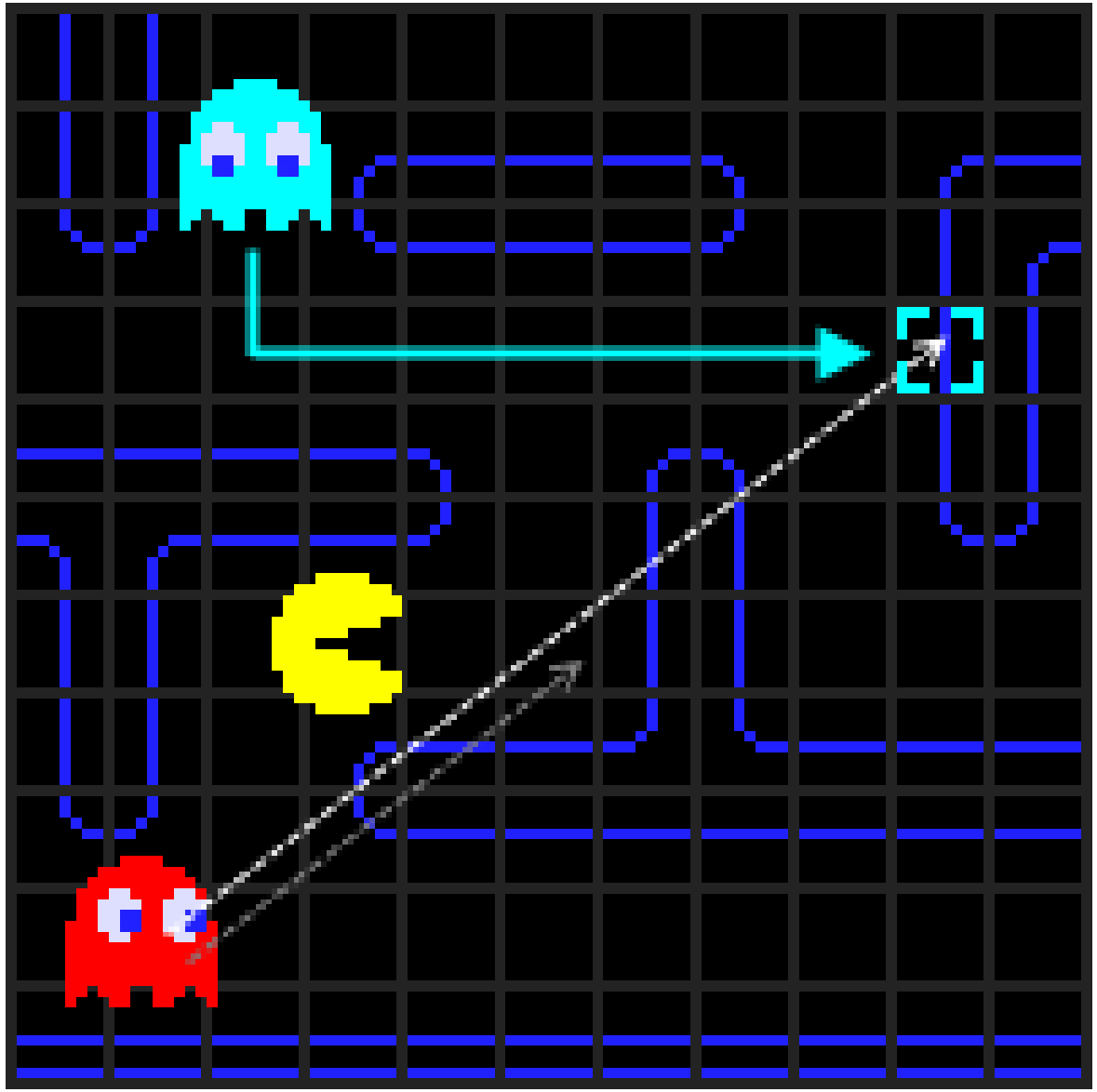
- Cieľová pozícia je 4 dlaždice pred hráčom
- Ak idete oproti duchovi tak ho môžete oklamať
- Chybné správanie – ak hráč ide smerom hore tak cieľ je 4 dlaždice pred hráčom a 4 dlaždice vľavo





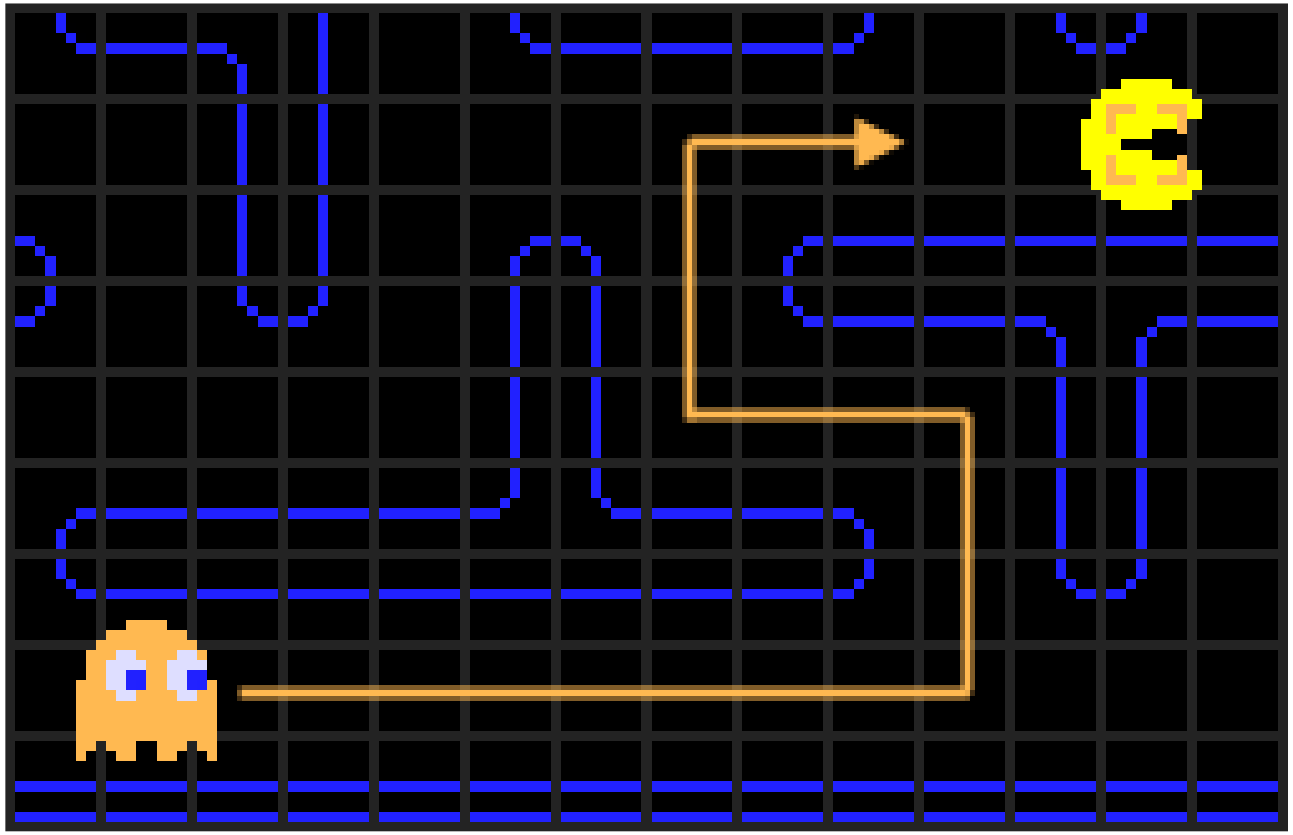
Modrý duch - Inky

- Ceiľová pozícia je vypočítaná ako 2x vektor z pozícií
  - 2 dlaždice pre hráčom v smere pohybu hráča
  - Pozícia červeného ducha
- Ak je „Blinky“ ďaleko, tak správanie je náhodné
- Ak je „Blinky“ blízko, tak naháňa hráča



Oranžový duch - Clyde

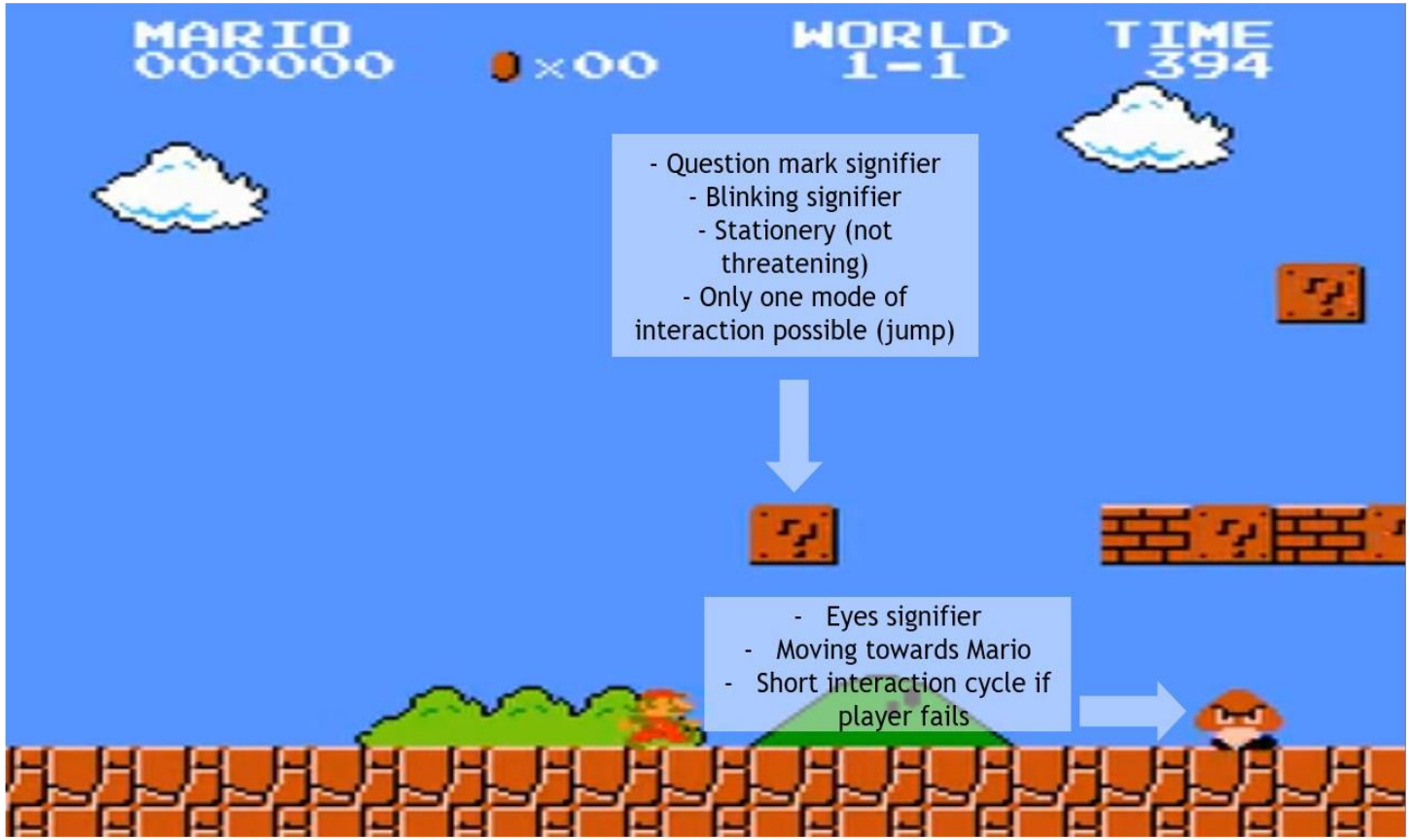
- Prepína medzi 2 módmi správania
  - Ak je od hráča ďalej ako 8 dlaždíc, tak sa správa ako Blinky (naháňa hráča)
  - Inak sa snaží dostať na domovskú pozíciu



Super Mario

Oboznámenie sa s prvkami

- Zaujímavé objekty
- Otáznik
- Hríb

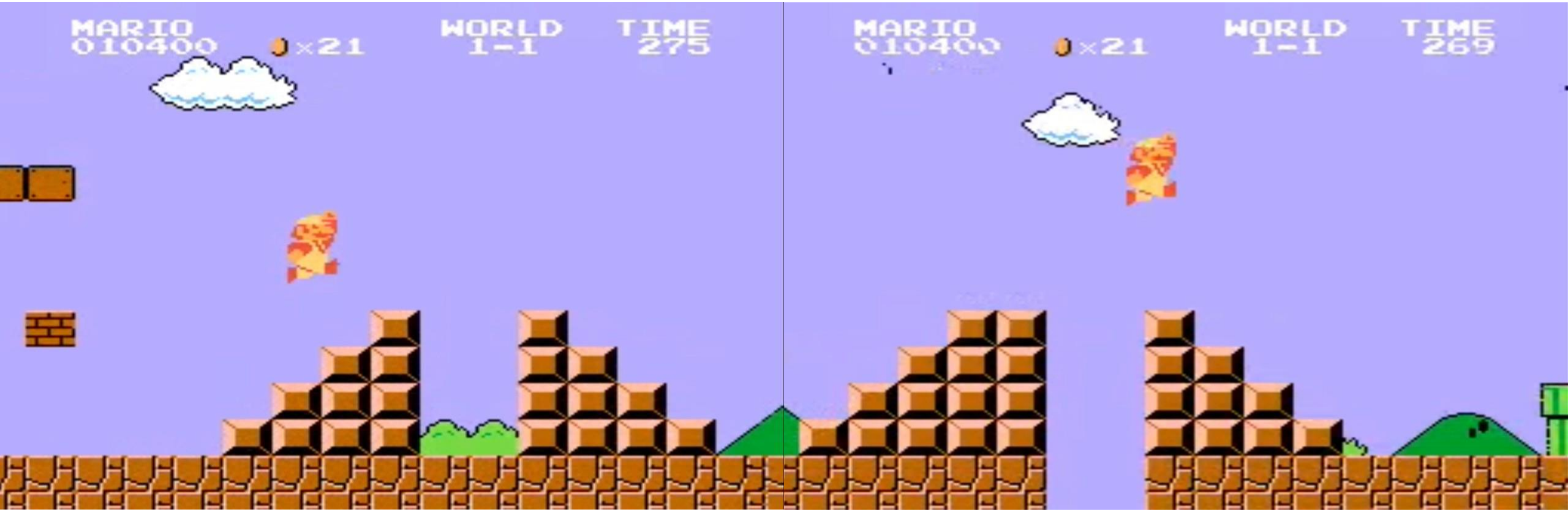


Oboznámenie sa s prvkami

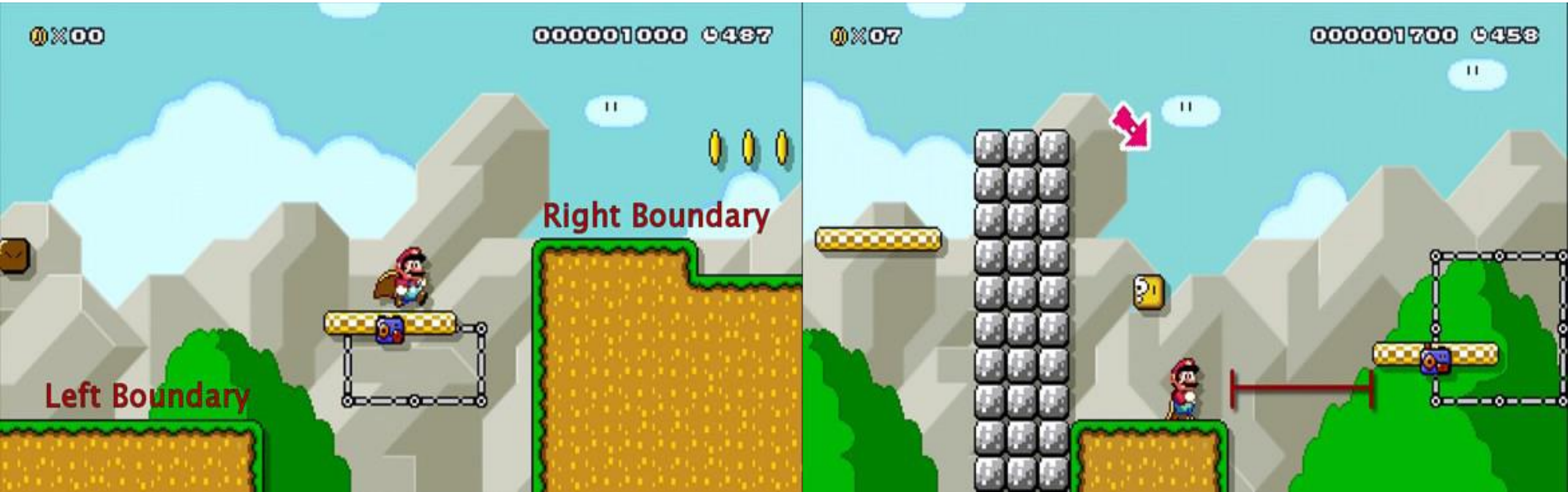
- PowerUps
  - Hviezdička
  - Hríb

Učenie sa v bezpečnom prostredí

Učenie sa v bezpečnom prostredí



Výzvy


×00
000001000
487
Right Boundary
Left Boundary
×07
000001700
458

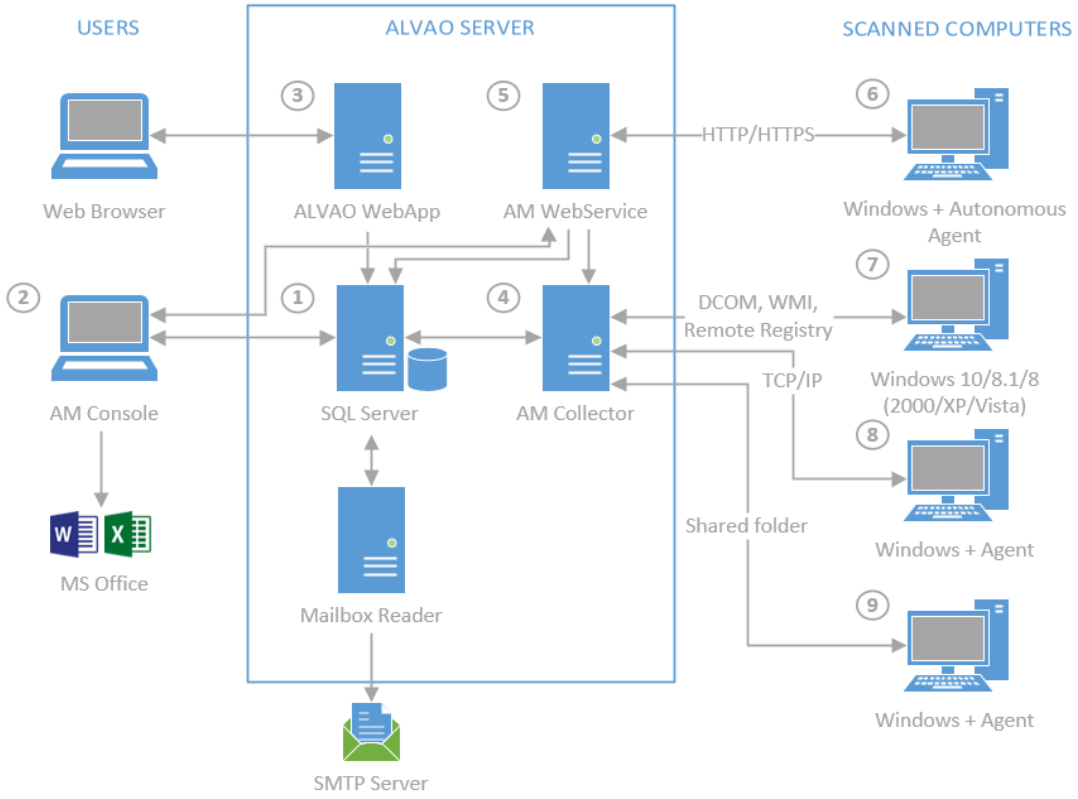
Výzvy

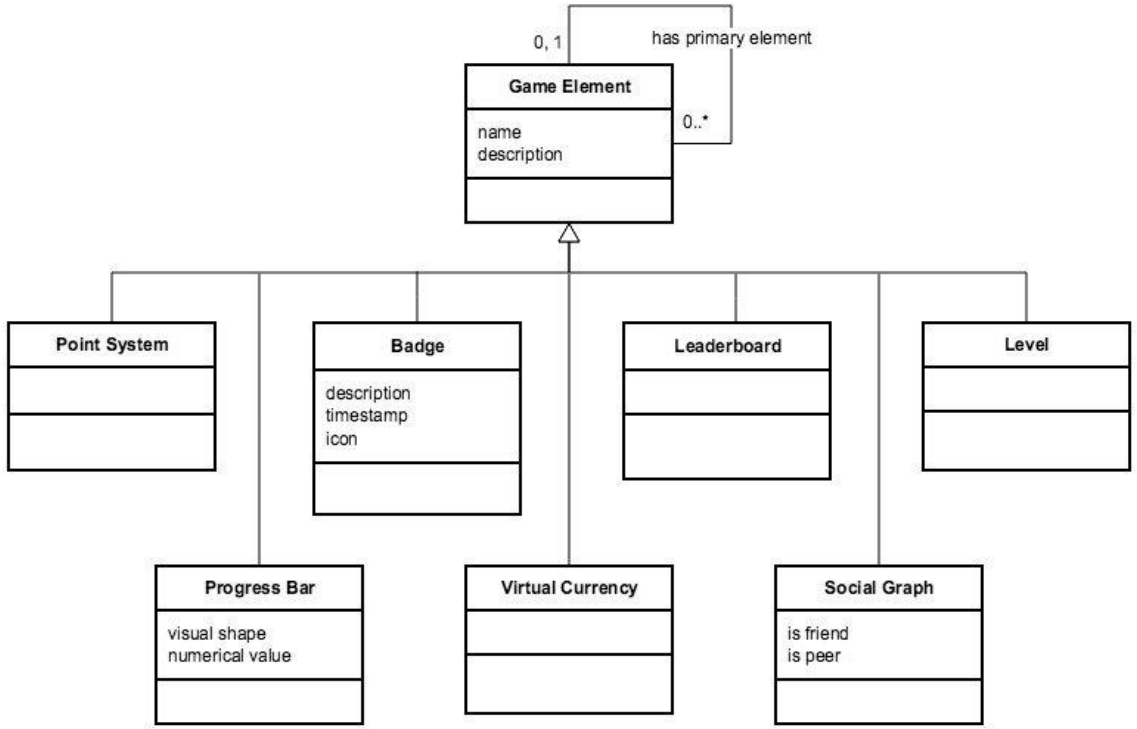
Návrh interaktívnej hry

Návrh

- Oblasti návrhu
  - Ciele – napr. pomocou MUDPY
  - GUI – napr. pomocou storyboards, mock-ups
  - Interakcia
    - Ovládanie aplikácie, odozva na udalosti ...
  - Herné mechanizmy
    - MDA, z používateľského pohľadu, z programátorského pohľadu
  - Architektúra
    - Všeobecný pohľad na riešenie

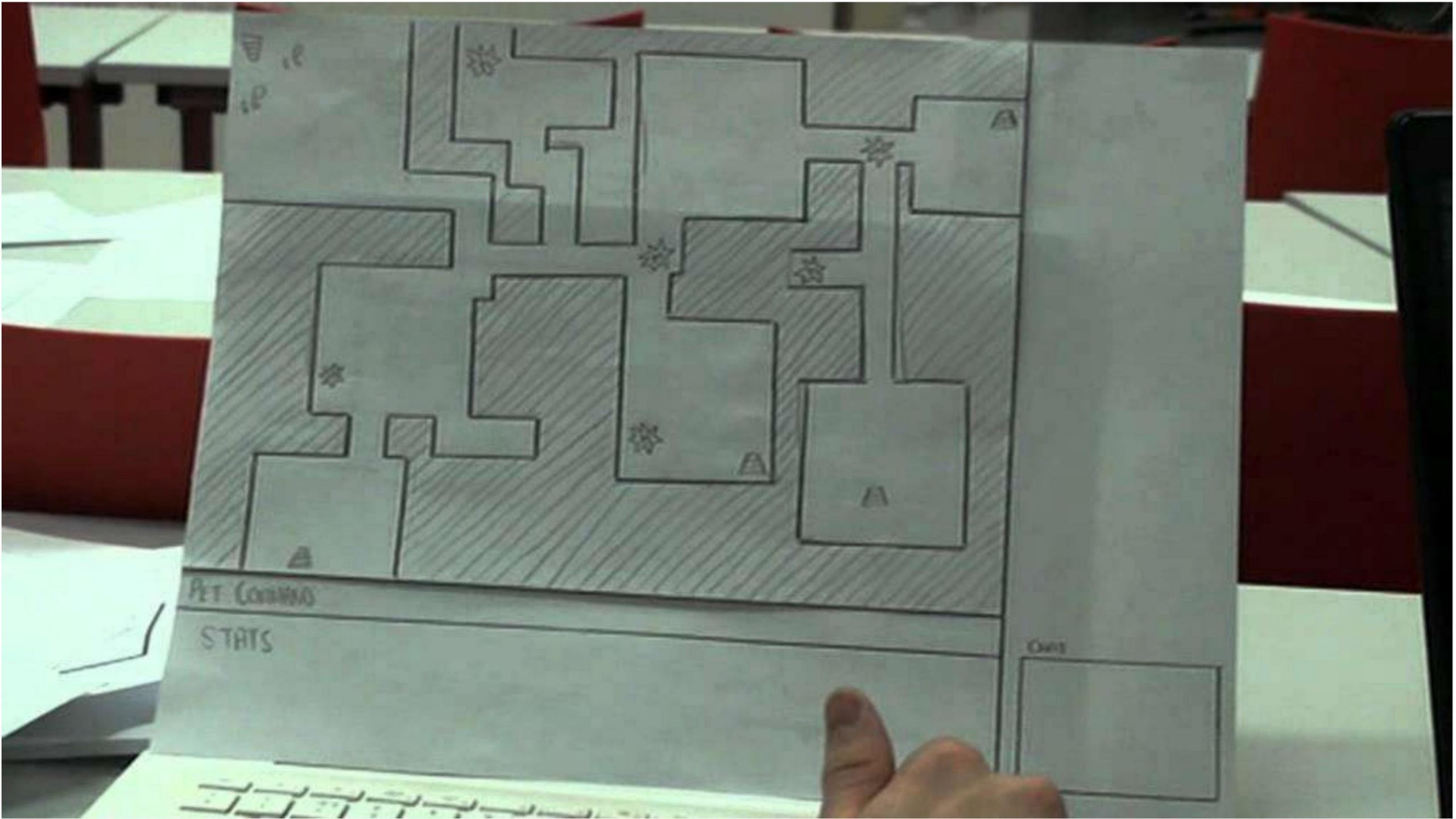
Architektúra





Vytvor prototyp

- Low-Fidelity
  - Pomocou ceruzky a papiera
  - Výhody
    - Lacné a rýchle
  - Nevýhody
    - Nesimuluje reálne odozvy počítača
- High-Fidelity
  - HTML, Mock-up, vytvorenie rýchleho prototypu


PET COMMAND
STATS

Mechanics-Dynamics-Aesthetics framework

Dizajnér a hráč

- Dizajnér
  - Mení pravidlá
  - Nepriamo ovplyvňuje hráča
- Hráč
  - Zmyslové vnímanie
  - Interakcia so systémom
  - Funguje podľa pravidiel
  - Emócie

Fázy

- Z pohľadu hráča
  - Pravidlá → Systém → „Zábava“
- Z pohľadu dizajnéra
  - Mechanika → Dynamika → Estetika

Mechanika

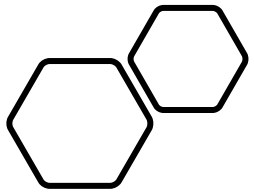
- Prvky, ktoré hráč neovplyvní, iba dizajnér/programátor
- Akcie, správanie a kontrolné mechanizmy
  - Pravidlá, grafika, dizajn levelu, elementy, a iné
- Cieľom je stanoviť základné správanie hráčov
- Drobné zmeny pravidiel môžu spôsobiť výrazné zmeny v dynamike

Dynamika (správanie)

- Správanie hráčov v hre
  - Podporovať hranie podľa pravidiel
  - Potláčať negatívne hranie - cheating
- Podpora nového správania
  - Kemping, blafovanie, agresívny alebo defenzívny štýl boja
- Individuálne charakteristiky hráča
  - Hráč sa sám rozhodne ako bude hrať

Estetika (emócie)

- Senzácia
- Fantázia a tvorivosť
- Príbeh
- Výzva
- Spoločenstvo
- Objavovanie



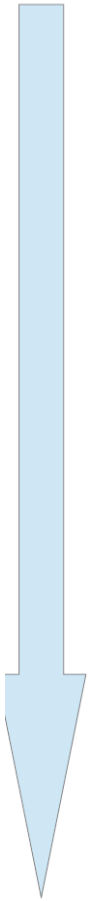
Game design vs Product design

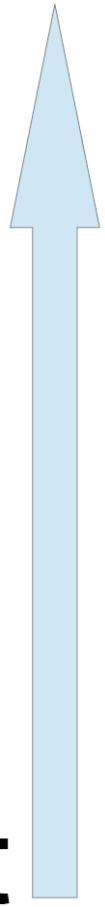
Dizajn zameraný na

produkt	zážitok
Funkcionalita	Aktivity ľudí, ich ciele
Výstup programu	Výsledky činnosti ľudí
Rozhranie, interakcia, použiteľnosť	Vnímanie, pamäť, emócie

Dizajn zameraný na zážitok

- Významný – má osobnú hodnotu
- Príjemný – zapamätateľný zážitok
- Praktický – funguje podľa očakávaní
- Použiteľný – jednoduchý na ovládanie
- Spoľahlivý – je prístupný a presný
- Funkčný – funguje podľa špecifikácie





Dizajn zameraný na produkt

Ako vhodne programovať

Názvy premenných a funkcií

- Výstižné názvy
  - Dočasné premenná – tmp, x ,y
  - Konštanta – Pí, Epsilon
  - Krátke názvy – uzol, xchg
- Používať anglické názvy
- Používať veľké a malé písmena - xchgFunction

Komentovanie

- Vysvetlenie kódu
- Nie príliš časté

Výpisy na overenie funkčnosti kódu

- printf(„Debug: opis chyby“);
- Správne umiestniť
- Logovanie

```
static void bubbleSort(int array[]){
    System.out.println("Debug: Spustam funkciu bubbleSort");
    for(int i = 0; i < array.length; i++){
        System.out.println("Debug bubbleSort: Prvy cyklus");
        for(int j = 0; j < array.length; j++){
            System.out.println("Debug bubbleSort: Druhy cyklus");
            if(array[i]<array[j]){
                System.out.println("Debug bubbleSort: vymena prvkov");
                int tmp = array[i];
                array[i] = array[j];
                array[j] = tmp;
            }
        }
    }
}
```

```
public class HelloWorld{


    static void bubbleSort(int array[]){
        System.out.print("Debug: Spustam funkciu bubbleSort");
        for(int i = 0; i < array.length; i++){
            for(int j = 0; j < array.length; j++){
                if(array[i]<array[j]){
                    int tmp = array[i];
                    array[i] = array[j];
                    array[j] = tmp;
                }
            }
        }
    }

    static void printArray(int array[]){
        System.out.print("Debug: Spustam funkciu printArray");
        System.out.print("Usporiadane pole: ");
        for(int x = 0; x<array.length; x++){
            System.out.print(" " + array[x]);
        }
    }

    public static void main(String []args){

        int array [] = {5,9,8,12,35,6,0,1,58,2,4};

        bubbleSort(array);
        System.out.print("Debug: Funkcia bubbleSort ukončená");
        printArray(array);
        System.out.print("Debug: Funkcia printArray ukončená");
     }
}
```

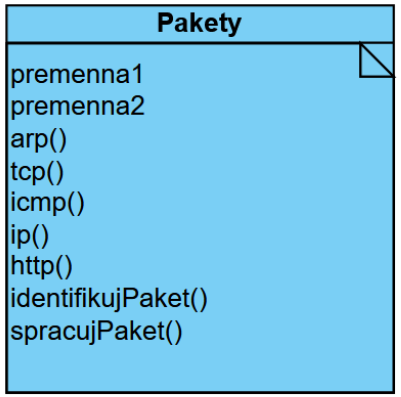
Ďalšie

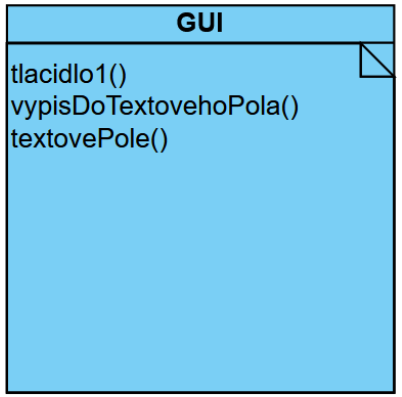
- Cykly
  - For vs While
- Udržiavať verzie kódu
- Odsadzovanie kódu
- Ak je to možné nepoužívať globálne premenné
- Testovanie
- Nie všetko v jednej funkcii

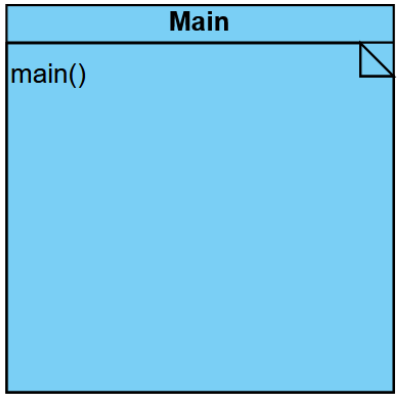
Použite viac súborov

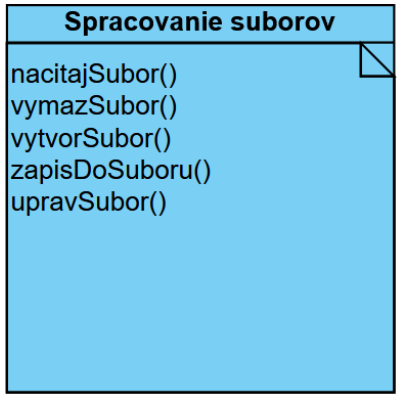
```
ArrayList<Integer> ICMP = new ArrayList<Integer>();
ArrayList<Integer> dlzka_ICMP = new ArrayList<Integer>();
ArrayList<Integer> headerlenght = new ArrayList<Integer>();
ArrayList<Integer> number = new ArrayList<Integer>();

ArrayList<String> IP_Sadress_ARP_request = new ArrayList<String>();
ArrayList<String> IP_Dadress_ARP_request = new ArrayList<String>();
ArrayList<String> IP_Sadress_ARP_reply = new ArrayList<String>();
ArrayList<String> IP_Dadress_ARP_reply = new ArrayList<String>();
ArrayList<Integer> cislo_request_ramca = new ArrayList<Integer>();
ArrayList<Integer> cislo_reply_ramca = new ArrayList<Integer>();
ArrayList<String> IP_adress_ARP_request = new ArrayList<String>();
ArrayList<String> IP_adress_ARP_reply = new ArrayList<String>();
ArrayList<Integer> vypis_request = new ArrayList<Integer>();
ArrayList<Integer> vypis_reply = new ArrayList<Integer>();
ArrayList<Integer> dlzka_ramcov_request = new ArrayList<Integer>();
ArrayList<Integer> dlzka_ramcov_reply = new ArrayList<Integer>();

ArrayList<Integer> ramce = new ArrayList<Integer>();
ArrayList<Integer> cislo_ramca = new ArrayList<Integer>();
ArrayList<Integer> velkost_ramca = new ArrayList<Integer>();
ArrayList<Integer> source_port = new ArrayList<Integer>();
ArrayList<Integer> velkost_hlavicky = new ArrayList<Integer>();
ArrayList<Integer> dest_port = new ArrayList<Integer>();
ArrayList<String> IP_Sadresy = new ArrayList<String>();
ArrayList<String> IP_Dadresy = new ArrayList<String>();

int []statistika = new int[8];
statistika[0]=0;
statistika[1]=0;
statistika[2]=0;
statistika[3]=0;
statistika[4]=0;
statistika[5]=0;
statistika[6]=0;
statistika[7]=0;

//sry za toto ale nejako mi blbo aj spoluziakov som sa pytal
if((pcap = Pcap.openOffline("trace-1.pcap", errbuf)) == null)
System.err.println("Opening file has failed");
PcapHeader header = new PcapHeader();
JBuffer buffer = new JBuffer(JMemory.Type.POINTER);
int pocitadlo = 1;
int value,max=0;
HashMap<String, Integer> hm = new HashMap<String, Integer>();

while ((buffer = pcap.next(header, buffer)) != null){
    int []zahumienka = new int[buffer.size()];

    for(int i=0;i<buffer.size();i++){
        zahumienka[i] = buffer.getUByte(i);
    }

    jta.append("\n"+"Ramec c."+pocitadlo);
    jta.append("\n"+"Dĺžka rámca poskytnutá paketovým driverom - " + buffer.size() + " B");
    if(buffer.size()<60) jta.append("\n"+"Dĺžka rámca prenášaného po médiu - 64 B" );
    else{
        int c = buffer.size() + 4;
        jta.append("\n"+"Dĺžka rámca prenášaného po médiu: " + c + " B");
    }

    jta.append("\nDestination MAC: ");
    for(int i = 0; i<6 ; i++){
        if(zahumienka[i]<16)
            jta.append("0");
        jta.append(Integer.toHexString(zahumienka[i])+" ");
    }
    //jta.append("\n");
```

```
premenna1
premenna2
premenna3
premenna4

arp(){
    kod
}

tcp(){
    kod
}

icmp(){
    kod
}

ip(){
    kod
}

http(){
    kod
}

identifikujPaket(){
    kod
}

spracujPaket(){
    identifikujPaket();
}

main(){
    spracujPaket();
}
```









Zhrnutie