1, 2, 3, кодирај! – Активности циклуса 4 - Пројект « Програмирање видео игре на платформи » - Етапа 5: Постављање лика на платформе

Доминантна дисциплина	Математика
Резиме	Ученици креирају свој лист пута програмирања за функционалност постављања лика « Играча » на платформе. Могуће је предвидети коришћење различитих алгоритама при чему би сваки од њих требало да користи условне исказе. Неки алгоритми могу да садрже багове. Ученици анализирају програм корак по корак да би елиминисали багове.
Појмови	« Језик » • Баг је грешка у програму. Баг може да потиче од грешке у концепцији алгоритма. « Добра пракса програмирања » : • Увођење паузе током извршења програма и налажење вредности варијабле олакшава решавање проблема бага. « Алгоритми» : • Алгоритам је нека врста метода који омогућује решавање проблема. Конструише се комбинацијом инструкција. • Више алгоритама може да омогући решавање истог проблема. • Неке петље, познате под именом "условне", се понављају док неки улсове не буде испуњен.
Материјал	За наставника: • Фајл *Скрач* Platformer_V03_demo.

Полазна ситуација

Наставник активира свој налог на *Скрачу* и отвара пројект Platformer_V04_demo. То је копија оног што је рађено на претходном часу, уз поједностављење пејсажа јер постоји само једна платформа у доњем делу сцене.Наставник објашњава да жели да укључи лик « Извиђача » који се налази у верзији V02. Разлика је у следећа два случаја:

- Ако наставник и ученици раде са *Скрачом* on-line, онда наставни показује како се користи « руксаг » *Скрача*. Отвара верзију пројекта V02, а затим отвара руксак, кликом на малу стрелицу која се налази у зони програмирања. Клиза лик « Извиђача » који ће од сада бити доступан на том месту. Затим се окреће верзији пројекта V04 и отвара руксак. Лик « Извиђача » се ту налази и довољно је да га пребацимо клизањем у зону ликова да би био инегрисан упројект са свим својим скриптама и костимима. Ученици се конектују на свој рачун Скрача и затим реализују исти рад укључујући и поједностављење пејсажа на једну једноставну платформу (хоризонтала раван). Региструју добијену програм пое именом Platformer_V04_име_групе.
- Ако наставник и одељење раде са локално инсталираним *Скрачом*, онда наставник показује како се уноси и износи лик. Отвара верзију пројекта V02 кликне на икону одабраног лика « регистрованог локлано као фајл ». Затим отвара верзију пројекта V04 уноси лик који је управо локално регистрован захваљујући икони « Унеси лик из фајла ».

Наставник захтева од ученика да реализују програм и да се припреме да кажу шта мисле о њему. Ученици покрећу проблем преласка лика преко плочица! Главни циљ овог часа је избегавање проблема и усредсређивање на програмирање ове функционалности менталне карте:

- Аватар се потавља на платформе.
- Основна карактеристика платформе је да аватар није у могућности да је пробије (одозго)

Ученици могу да идентификују други проблем који се односи на почетак падања « Извиђача » пре него што је пејсаж поплочан. Наставник то наглашава па ако је потребно одељење може да дода одговарајуће функционалности у менталну карту програма на следећем час:

- Ниво: информисати аватара о његовој припреми.
- Аватар: појављује се и покреће лпошто је пејсаж поплочан.

Креирање мапе програмирања (по паровима)

Наставник предлаже ученицима да креирају своју мапу пута. Може повремено да умерава размишљање неко пара, али не интервенише на други начин.L’enseignant peut ponctuellement guider la réflexion d’un binôme, mais il n’intervient pas outre mesure. Ученици прво праве листинг задатака а затим постепено прелазе на програмирање.

Реализација програмирања (по паровима)

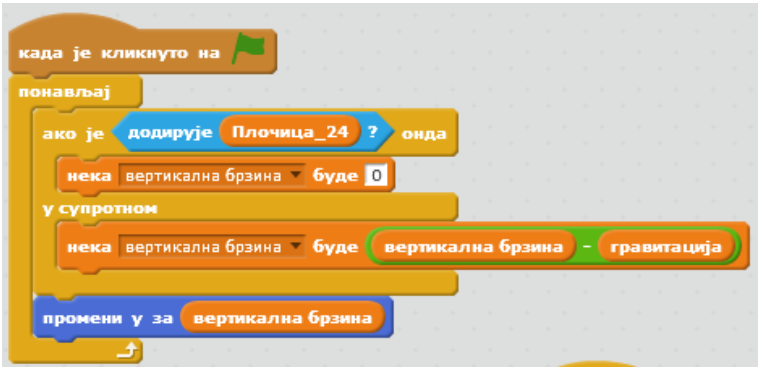
Ученици се самостално ангажују на реализацији задатака с направљене листе, а наставник иде од групе до групе и размењује мишљење с ученицим ау вези достигнутог нивоа активности. Подсећа их да редовно меморишу оно што су урадили на пројекту.

Педагошке напомене:

- Аутономно формирана листа задатака по групама је извор различитости у приступу али и извор потешкоћа у прграмирању. Зато се и ограничавамо само на програмирање једне једине доста једноставне функциоанлности.
- Наставник позива ученике да се инспиришу с листама путева програмирања које су користили на претходним часовима.
- Може да пита ученике која им је варијабал и функција потребна.

Заједничко представљање

Наставник идентификује групу која се суочила с неким текучим проблемом и предлаже им да прикажу свој програм у *Скрачу*. Група представља свој проблем одељењу у покушају да нађу заједнчко решење.
Узмимо конкретан пример код који се односи на « Играча » који се при паду није зауставио на платформи него се зарио више пиксела у њу и остао заглављен!
Ученици модификују подпрограме, који контролишу пад играча, увођењем условне инструкције попут ове:

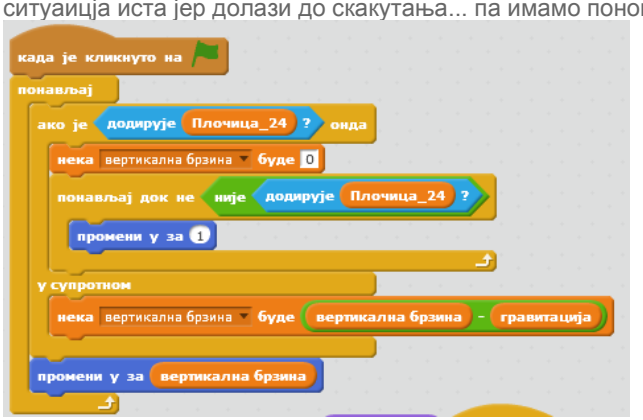


Зашто баш овај проблем? Очигледно је да « Играч » при завршетку свог пада детектује плочицу, али уместо да се заустави на њеној површини он се у њу зарије. Да би боље видели шта се дешава можемо да захтевамо приказ вредности варијабле « вертикална брзина » на сцени, и/или увођење мале паузе пре вертикалних померања, « чекај 0.1 секунди ». Закључујемо да је продирање у плочицу последица дискретизације померања, јер « Играч » пада, на пример, по 10 пикслеа па отуда и ситуација да се зарије у плочицу. Пре овог померања, он не додирује плочицу, дакле потпуно је нормално да он падне за « вертикалну брзину » пиксела. Од сада он додирује плочицу, а његова вертикална брзина узима вредност нулу. Логично је да се зарије у плочцу. Први закључак је да је програмом урађено оно што је од њега захтевано. Потребно је да нађемо начин да се « Играч » поново нађе на плочици.

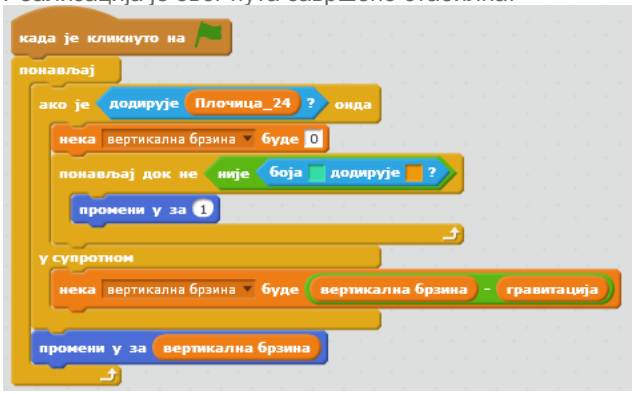
Научна напомена:
Овај проблем настаје само ако су ученици одлучили да своје « Плочице » постављају « клонирањем ». У случају да примене метод « печатирања », судари се регулишу само детекцијом боје. Печатирање је флуиднић метод за регулацију судара.

Могуће је предвидети више метода, или алгоритама, који се могу тестирати заједнички уз вођење наставника ако је то неопходно:

- Вертикалној брзини, при додиру плочице, можемо дати неку позитивну вредност (на пример 1) уместо 0. « Играч » ће се тад подићи увис. Инплементација овог предлога омогућује « Играчу » да перманентно скакуће. Да ли нам се то свиђа или не... то је већ ствар укуса. Ово, у свакомслучају, представља проблем за касније програмирање функционалности скока с платформи, јер скок може да се деси само када лик додирне површину платформе, што нијекада се лик подигне.
- Можемо, када је плочица додирнута, да вертикалној брзини дамо врдност 0 као у почетном програму који су правили ученици. Међутим, да би се « Играч » подизао пиксел по пиксел потребно је мењати претходно дефинисану вредност варијабле за ординату Y док он не додирне плочицу. Ипак, и овде је ситуаиција иста јер долази до скакутања... па имамо поново проблем скокова као у претходном алгоритму.

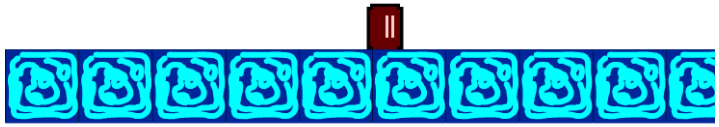


- Неки други алгоритам користи сензоре контакта између боја. Тако се « Играч » налази на линији периферије црног пиксела, и на црвеном пикселу ако се зарије у плочицу. Ако користимо горњи предлог условљавајући пењање пиксел по пиксел у контакту између црвене боје у дубини и боје површине плочица, онда ће се « Играч » пењати само дуж линије црног пиксела. Када додирне плочицу (дакле његова вертикална брзина ће бити нула), и више нема пењања. Реализација је овог пута савршено стабилна.



Овај алгоритам садржи могућу корекцију задатка у вези програмирања сцене.
Ако има времена друга група ученика може да представи своје потешкоће и покуша да их заједнички реши. Приступ је исти: анализира се програм, уводи пауза, пприказује вредност неке варијабле, предлажу алгоритми за тестирање.

Предвиђа се и време потребно да групе поправе свој програм и узму у обзир оно што је научено током заједничког представљања.
Игра, на екрану, приказује лик који је способан да падне али и да се стабилизује ако је дошао у контакт с неким елементом платформе:



Закључак

Наставни усмерава одељење ка формулисању следећег закључка:

- *Алгоритам је мтод који омогућује решавање неког проблема. Конструише се комбинацијом инструкција.*
- *Један исти проблем је могуће решити коришћењем различитих алгоритама.*
- *Баг је грешак у програму. Баг може да буде последица грешке у концепцији алгоритма.*
- *Увођење пауза при извршењу неког програма и посматрање вредности варијабли олакшава решење багова.*
