Pedagogická fakulta
Faculty of Education
Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky

Diplomová práce

Ověřování výukových materiálů pro výuku informatického myšlení v prostředí Unreal Engine 4

Vypracoval: Bc. Dominik Paclt
Vedoucí práce: doc. PaedDr. Jiří Vaníček, Ph.D.

České Budějovice 2022

Anotace

V rámci mého bakalářského studia byla vytvořena sada pro výuku programování ve vývojářském prostředí Unreal Engine 4. Hlavním cílem této práce je ověřit, zda je práce pro výuku programování vhodná, či nikoliv.

Teoretická část se zabývá základním popisem prostředí Unreal Engine 4. Dále se věnuje popisu termínu „informatické myšlení“ a jeho složek. Následně je rozebírán princip Akčního výzkumu, skrze který je práce testována a posouzení, zda úlohy podporují informatické myšlení.

Praktická část obsahuje výsledky z akčního výzkumu jak z pohledu splnění cílů výuky u jednotlivých lekcí, tak z pohledu vedení k informatickému myšlení.

V příloze jsou obsaženy Testované pracovní listy pro žáky a samotné projekty řešené žáky v Unreal Engine 4.

V rámci projektu GAJU STEM má tato práce Uplatnění především v technickém oboru, neboť jednotlivé lekce podporují algoritmické přemýšlení, což je jedním ze základů technických oborů. Dále jsou v úlohách silně patrny mezipředmětové vztahy, především pak do matematiky a fyziky, neboť v úlohách je patrna jednoduchá matematika a celé prostředí je situováno do simulace reálného světa (z pohledu Fyziky).

Klíčová slova

Výuka programování, věk žáka, Informatické myšlení, Unreal Engine 4, Epic Games Launcher, herní prostředí

Annotation

As part of my bachelor's degree, a set for teaching programming in the Unreal Engine 4 development environment was created. The main goal of this work is to verify whether the work is suitable for teaching programming or not.

The theoretical part deals with the basic description of the Unreal Engine 4. It also deals with the description of the term "computer thinking" and its components. Subsequently, the principle of Action Research is discussed, through which the work is tested and assessment of whether the tasks support computer thinking.

The practical part contains the results of action research both in terms of meeting the objectives of teaching in individual lessons, and in terms of leading to computer thinking.

The appendix contains Tested worksheets for students and the projects solved by students in u nreal Engine 4.

Within the GAJU STEM project, this work is u sed mainly in the technical field, as the individual lessons support algorithmic thinking, which is one of the foundations of technical fields. Furthermore, interdisciplinary relationships are strongly evident in the tasks, especially in mathematics and physics, because simple mathematics is evident in the tasks and the whole environment is situated in the simulation of the real world (from the point of view of Physics).

Keywords

Teaching programming, pupil age, Informatics thinking, u nreal Engine 4, Epic Games Launcher, game environment

Poděkování

Děkuji doc. PaedDr. Jiřímu Vaníčkovi, Ph.D. za možnost realizace této práce pod jeho vedením, za rady a informace, které mi poskytl. Především za důvěru a trpělivost během realizace mé práce. Dále bych rád poděkoval Základní škole Choustník, zvláště panu řediteli Mgr. Karlu Zvěřinovi za možnost spolupráce a umožnění vedení kroužku, na kterém jsem svou práci mohl testovat.

Prohlášení

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury u vedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě – v úpravě vzniklé vypuštěním vyznačených částí archivovaných Pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou Univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s u vedeným u stanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Dominik Paclt
25.06.2022

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Pedagogická fakulta

Akademický rok: 2020/2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Dominik PACLT**
Osobní číslo:	**P20818**
Studijní program:	**N7503 Učitelství pro základní školy**
Studijní obor:	**Učitelství fyziky pro 2. stupeň základních škol** **Učitelství informatiky pro 2. stupeň základních škol**
Téma práce:	**Ověřování výukových materiálů pro výuku informatického myšlení v prostředí Unreal Engine 4**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování

Pro prostředí Unreal Engine 4 pro tvorbu počítačových her byla vyvinuta sada vzdělávacích materiálů, která je určena k výuce základů programování. Ověřováním při výuce lze odhalit možnosti a nedostatky těchto materiálů a následně navrhnout úpravy s cílem tyto materiály vylepšit tak, aby v žácích podporovala rozvíjení některých složek informatického myšlení.

V teoretické části práce student představí pojem informatické myšlení, jeho jednotlivé složky, dotýkající se tématu, a vybere ty ze složek, které mohou být programováním her rozvíjeny. Student stručně představí prostředí Unreal Engine 4. Dále student popíše metodu ověřování výuky akčním výzkumem. Bude analyzovat danou sadu úloh pro výuku programování v tomto prostředí z hlediska vybraných složek informatického myšlení, zda jsou nebo nejsou v úlohách přítomny.

V praktické části student ověří samotnou sadu úloh ve výuce. Provede řádný akční výzkum a nashromážděné poznatky bude analyzovat z pohledu, nakolik se dařilo naplnit cíle výuky a také rozvíjet informatické myšlení žáků. Následně poskytne návrhy na vylepšení daných úloh a ty realizuje.

Rozsah pracovní zprávy:	**60**
Rozsah grafických prací:	**CD ROM**
Forma zpracování diplomové práce:	**tištěná**

Seznam doporučené literatury:

1. PAVELKOVÁ, A. Akční výzkum v pedagogickém prostředí [online]. 2012. Dostupné z: https://is.muni.cz/th/yxbkp/Akcni_vyzkum_v_pedagogickem_prostredi_.pdf
2. WALTEROVÁ, E. Akční výzkum v podmínkách české školy [online]. Dostupné z: http://capv.cz/images/sborniky/1995/walterova.pdf
3. LEE, J. Learning Unreal Engine Game Development [online]. Packt Publishing Limited, 2016.
4. ČÁPKA, David. ITnetwork.cz [online]. 2019. Dostupné z: https://www.itnetwork.cz/it-e-learning.
5. Epic Games. Unreal Engine 4 Documentation [online]. 2019. Dostupné z: https://docs.unrealengine.com/en-US/index.html
6. JELÍNEK, J. Fundamenty informatického myšlení [online]. 2008. Dostupné z: https://dspace.cuni.cz/handle/20.500.11956/16562
7. KREJSA, J. Výuka základů programování v prostředí Scratch [online]. České Budějovice, Dostupné z: http://theses.cz/id/b5f11x/

Vedoucí diplomové práce: **doc. PaedDr. Jiří Vaníček, Ph.D.**
Katedra informatiky

Datum zadání diplomové práce: **6. dubna 2021**
Termín odevzdání diplomové práce: **30. dubna 2022**

L.S.

doc. RNDr. Helena Koldová, Ph.D.
děkanka

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 6. dubna 2021

Obsah

1. Úvod

1.1. Cíl práce

Unreal Engine 4

Tato část se zaměří na obecný popis prostředí Unreal Engine 4. V této části bude popsáno zaměření Unreal engine 4, bude představen systém Blueprint, který nahrazuje textový programovací jazyk a budou představeny nejzákladnější třídy systému (především třída herec).

Informatické myšlení

V této části práce bude představen pojem informatické myšlení. Při tom budou představeny i jeho jednotlivé složky, které budou demonstrovány na napřed obecných příkladech a následně na příkladu z vývoje her.

Analýza úloh

V této části proběhne analýza úloh z práce "Tvorba projektů pro výuku programování v prostředí Unreal Engine 4". Analýza proběhne z hlediska toho, zda úlohy obsahují podněty pro rozvíjení jednotlivých složek informatického myšlení, či nikoliv.

Akční výzkum

V této části bude popsán způsob prověřování výuky Akčním výzkumem. Bude představen popis zkoumání včetně východisek a podmínek, které musí splňovat.

Ověřování úloh

V této části budou rozepsány výsledky získané během výzkumu, který budu provádět v průběhu ověřování sady úloh ve výuce. Nashromážděné poznatky budou analyzovány z pohledu toho, nakolik se dařilo naplnit cíle výuky stanovené v dané práci u každé lekce a zda se dařilo v žácích rozvíjet informatické myšlení.

1.2. Metoda práce

Více než sedm let před samotným počátkem mé práce jsem již v prostředí UE4 pracoval a vytvářel v něm software. Z toho důvodu budu toto prostředí popisovat za pomoci textů, které jsem sepsal dříve, jenž jsou opřeny o literaturu.

K popisu informatického myšlení budu analyzovat odborné zdroje, pojednávající o dané problematice. Jako demonstrační příklady využiji výstupy od žáků, kteří v roce 2020 dokončili zde zkoumanou sadu úloh a kteří v prostředí UE4 pracují do dnes (rok 2021).

Analýza úloh z pohledu obsahu složek rozvíjejících informatické myšlení bude selektivní, s podporou odborné literatury. Během analýzy proběhne selekce podnětů pro podporu rozvoje informatického myšlení.

K popisu Akčního výzkumu budu studovat odbornou literaturu, ze které v dané části budu citovat.

Samotné ověřování úloh proběhne za využití akčního výzkumu formou vedení kurzu programování v Unreal Engine 4, kde budou výstupy celého plánu zkoumány. Zjištění, zda proběhlo zlepšení informatického myšlení žáků bude provedeno na základě srovnávacích testů.

2. Unreal Engine 4

V rámci své bakalářské práce jsem provedl širší popis prostředí Unreal Engine 4. Z toho důvodu při obecném popisu tohoto prostředí využívám textů, které jsem již dříve napsal.

„V následujících kapitolách popíšu termín herní Engine, představím UE4 a jeho základní součásti. Dále popíšu systém Blueprint, u něhož je kód tvořen graficky, a využívám jej ve své sadě úloh.“ [1]

2.1. Herní engine

„Herní engine je software, který poskytuje propojené nástroje a programy, jež pomáhají při konstrukci hry. Může obsahovat grafické programy, programovací programy, programy pro modifikaci zvuku či simulační programy. Díky tomu dává znatelný náskok oproti jiným způsobům vytváření her.“ [1]

2.2. Úvod k prostředí Unreal Engine 4

„Unreal Engine 4 je v dnešní době jeden z nejpopulárnějších systémů na trhu. Můžou za to jeho možnosti pro doladění grafické stránky přímo ve hře (především pak systém světla a stínů, který je v dnešní době jedním z nejpokročilejších systémů), jeho rozsáhlé přizpůsobitelnosti pro uživatele, schopnostem převést hru na velké množství platforem a v neposlední řadě možnost zdarma vyvíjet jakýkoliv produkt (až do chvíle, kdy produkt začne generovat zisk).“[1]

„Programovacím jazykem UE4 je C++. Tímto jazykem je prostředí naprogramováno, i se s ním mohou za pomoci prostředí „Microsoft Visual Studio“ programovat produkty. Krom jazyka C++ je v UE4 ještě jeden způsob programování, systém Blueprint. Jde o systém vizuálního skriptování, který umožňuje rozšířit funkčnost kódu pomocí vizuálního skriptovacího jazyka (uzly, propojené čárami). Tento systém, si přiblížíme v následujících kapitolách.“ [1]

„Další výhodou je volný přístup ke zdrojovému kódu Unreal Engine, který dává uživatelům možnost přetvořit samotný systém a vytvořit tím téměř cokoliv i nad rámec původních možností UE4.“ [1]

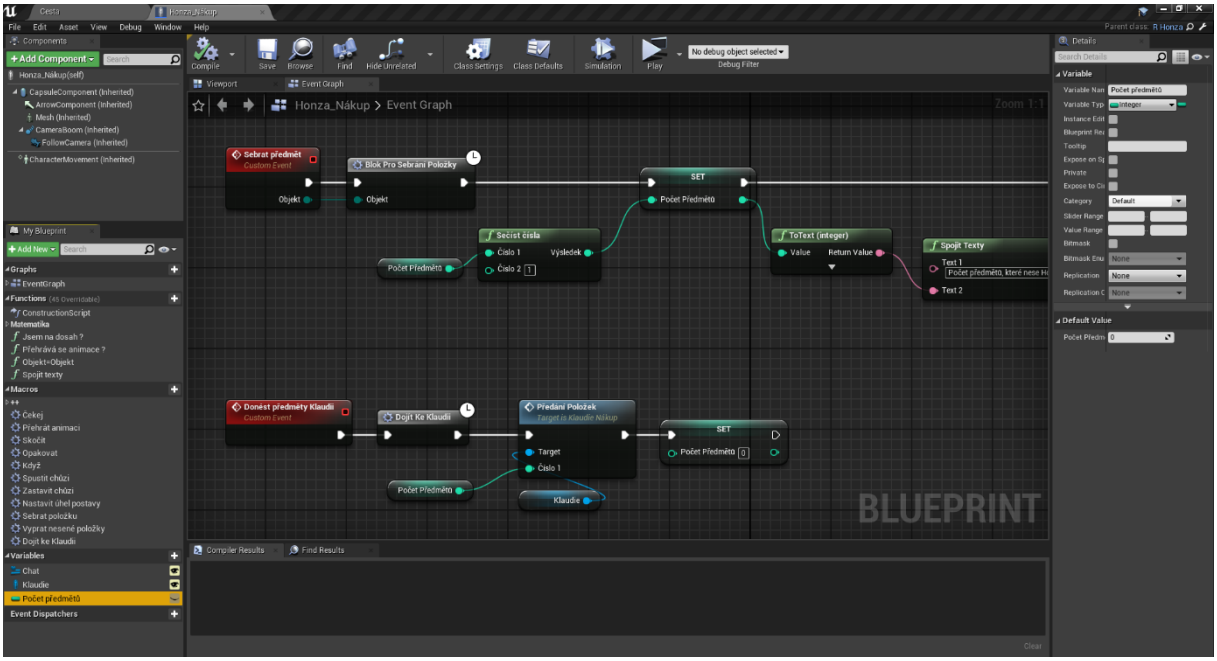
„Unreal Engine se skládá z několika součástí, které spolupracují na řízení produktu. Některé z nich se věnují vzhledu finálního produktu (př. „Zvukový engine“ nebo „Grafický engine“) jiné se věnují chování finálního produktu (př. „Fyzikální engine“, „Systém umělé inteligence“, či „Systém Blueprint“).“ [1]

„Tyto jednotlivé součásti jsou vytvořeny tak, že pro práci v jedné části není potřeba znát části jiné. Pokud je však potřeba testovat např. kód postavy, je již vhodné pro názornost ovládat základy práce s mapou.“ [1]

2.3. Systém Blueprint

„Jedná se o funkci přidanou až v Unreal engine ve verzi 4. Jak již bylo řečeno, jde o systém vizuálního skriptování, který umožňuje rozšířit funkčnost kódu pomocí vizuálního skriptovacího jazyka. Systém programování by se dal přirovnat k vytváření vývojového diagramu.“ [1]

„Díky této schopnosti není nutné psát kód, čímž částečně odpadá potřeba kontrolovat syntaktické chyby. Skvělou funkcí Blueprint je, že můžete vytvářet proměnné, funkce i makra kliknutím na tlačítka „+ Function“, „+ Macro“ a „+ Variable“. Není proto potřeba vytvářet kód pro deklarování. S tím je spojena i výhoda, že takto vytvořené proměnné, funkce a makra jsou zobrazena v příslušných seznamech, a jsou viditelná po celou dobu skriptování v dané třídě (daném objektu).1]“ [1]



OBRÁZEK 1 UKÁZKA SYSTÉMU BLUEPRINT

„Výrazný nástroj pro podporu vývoje v systému Blueprint je i možnost spustit testování téměř v každém momentu hry. Během testování je možné vidět průběh událostí a hodnot vlastností vizuálně přímo v kódu a v reálném čase.“ [1].

„V tomto systému lze krom tříd Actors programovat samostatné funkce i Makra, které lze následně používat globálně (s určitým omezením). Dále v něm lze modifikovat vlastnosti

samotné úrovně, nastavovat třídy, starající se o celkový průběh výsledného produktu (jako je například třída „Player Controller“, starající se mimo jiné o přijímání vstupů na uživatele).V neposlední řadě s ním jde programovat vizuální rozhraní pro uživatele „HUD“, na něž lze zobrazit tlačítka, obrázky a jiné části, sloužící pro komunikaci s uživatelem.“ [1]

3. Informatické myšlení

Jedná se o způsob myšlení, při kterém identifikujeme problém, který je potřeba vyřešit, na což jej s pomocí informatických aparátů řešíme. [2] [3]

Další možné vyjádření informatického myšlení může být takový, že je to proces, kdy přemýšlíme o problémech takovým způsobem, jakým by je řešily počítače. [4]

Při řešení takovéhoto problému je podstatné na začátku co nejpřesněji definovat problém, který řešíme, určit, jaké okolnosti vedly ke vzniku problému a vymyslet kostru řešení problému. Tyto řešení se při tom snažíme vytvářet tak, abychom je mohly znovu využít při řešení jiného problému. [2] [3]

Cílem při vedení k informatickému myšlení je motivovat žáky ve vymýšlení vlastních řešení problému místo toho, aby se striktně držely již vytvořených postupů někoho jiného. Tím se žáci, učí kreativitě, či schopnostem logického uvažování a abstrakce. Při tom se žáci, učí pracovat s chybou a brát si z ní ponaučení. Dále je vhodné, když žáci pracují ve skupinách, aby se naučili vzájemné spolupráci. Tím mezi nimi vzniká dialog, který napomáhá pochopení daních problémů.[3] [4]

Informatické myšlení se skládá ze čtyř složek (dekompozice, vzory a sekvence, abstrakce a algoritmizace) a tří postupů (generalizace, optimalizace a evaluace), které si v následujících odstavcích představíme [4]:

3.1. Dekompozice

3.1.1. Popis

Při využití dekompozice, rozkládáme větší a na první pohled složitější problém na několik dílčích. Přitom dbáme na zachování správných vztahů mezi nimi. Takovéto rozdělení nám u možní snadněji zvládnout problém jako celek, snadněji jej řešíme a lépe mu rozumíme. [2] [3] [4]

3.1.2. Příklady

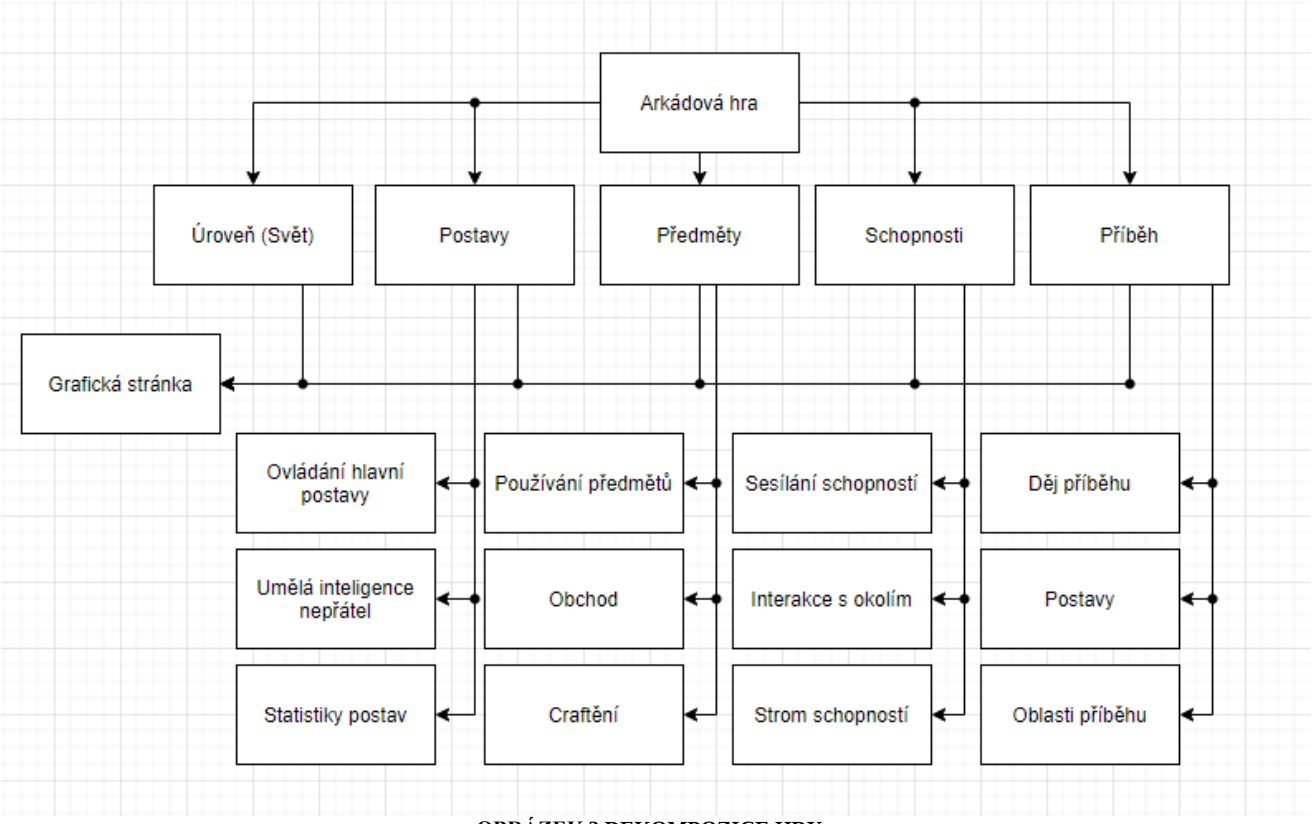
Příklady využití dekompozice z reálného života mohou být různé. Pro jednou se můžeme chtít dovtípit, jak funguje nějaký objekt, jako je například kolo. Pro jeho celkové pochopení si jej

často rozdělujeme na dílčí části (převodovka, brzdy, plášť, …), u kterých se snažíme zjistit jejich funkci. Ve chvíli, co chápeme všechny části, zkoumáme jejich vzájemné propojení, až nakonec pochopíme, jak funguje kolo jakožto celek. [3]

Dalším příkladem může být rozbor nějaké činnosti, například příprava snídaně. Při této přípravě rozdělíme celkovou činnost na dílčí činnosti (příprava toustu, u dělání čaje, u vaření vejce, …). Každá z těchto činností může být téže dále rozdělena do dílčích kroků (např. příprava čaje: napuštění vody, ohřev vody, výběr příchutě, nalití do hrnečku). [5]

3.1.3. Příklad pro vývoj her

Nyní demonstruji dekompozici na tématu této práce čili vývoji hry. Například pro jednoduchou arkádovou hru může dekompozice vypadat obdobně jako na obrázku níže:



OBRÁZEK 2 DEKOMPOZICE HRY

Obrázek výše dělí Arkádovou hru do 5 částí, které jsou následně dále děleny. Takováto dekompozice není zdaleka dokonalá ani kompletní, u ž jenom pro to, že zde nejsou vyznačeny spojitosti mezi dílčími částmi hry, nicméně to ani nebyl záměr, neboť jsem chtěl pouze demonstrovat dekompozici na jednoduchém grafu. Kompletní dekompoziční graf by byl exponenciálně složitější a násobně nepřehlednější, neboť takováto hra by obsahovala mnoho dalšího (př.: widgety, ukládání postupu, popřípadě multiplayer, …).

Kvůli takovémuto členění mohou například různí lidé pracovat na různých částech hry nezávisle na sobě (stačí navrhnout vhodná komunikační vlákna). To vše však za předpokladu, že předem provedou podrobnou dekompozici a domluví se na klíčových aspektech hry. [5]

„S žáky z kroužku, se kterými jsem testoval svou bakalářskou práci, stále v daném kroužku pokračujeme, avšak nyní se zaměřujeme na zhotovení funkčních her. Takto jsme vytvořily už 4 funkční hry a z této zkušenosti mohu potvrdit slova v odstavci výše, která apelují na dekompozici před začátkem vývoje. Je to opravdu důležité.“

3.2. Vzory a sekvence

3.2.1. Popis

Hledání vzorů, aneb částí, které se opakují. Při identifikaci takovéhoto vzoru můžeme složitý problém řešit mnohem efektivněji, neboť můžeme využít již hotový postup pro jeden vzor a ten následně použít znova. Při tom můžeme využívat cykly. Chceme-li nacházet vzory v problémech, musíme pátrat po věcech, které jsou v každém problému stejné, nebo velmi podobné. [6] [7]

3.2.2. Příklady

Příkladem může být například stavění cihlové zdi. Poté, co provedeme přípravu ke stavbě (příprava nářadí, cihel malty, …), je celý proces zasvěcen tomu, že dělník:

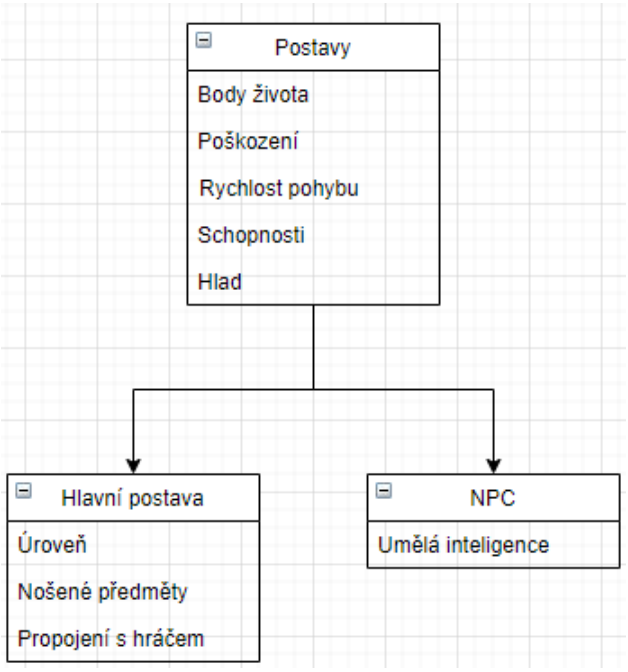
- Zvedne cihlu
- Nanese maltu
- Usadí cihlu

Takovýto postup opakuje až do chvíle, než je postavena celá zeď. Případ je sice velmi zjednodušen, neboť během stavby se mohou vyskytnout problémy, které je třeba řešit (zlomená cihla, zarovnání malty, nedostatek materiálů, …), avšak tyto záležitosti lze ošetřit přímo v algoritmu s pomocí optimalizace, o kterých píšu dále.

Dalším příkladem může být kresba květiny v geometrii. po uvědomění si, že jednotlivé lístky představují geometrické tvary můžeme kreslit jeden tvar stále dokola, akorát se změnou lokace a rotace, až nakonec může vzniknout obrazec, připomínající květinu.[8]

3.2.3. Příklad pro vývoj her

Jelikož jsem u dekompozice použil příklad arkádové hry, použiji i zde příklad na arkádovou hru. V arkádové hře jsou z pravidla dva typy postav, a to sice postavy ovládané hráčem (hlavní postavy) a postavy ovládané počítačem (NPC).



OBRÁZEK 3 DĚLBA POSTAV

Nyní si položme otázku: „Mají tyto dvě skupiny postav nějaké společné rysy (vzory)?“. po zkoumání můžeme určit, že oba typy postav budou mít obecné vlastnosti, jako je množství bodů života, po jehož ztrátě postavy zemřou, poškození, které jsou schopné udílet jiným postavám a mnohé další. a co vlastnosti, které nejsou společné? NPC naproti hlavním postavám nemusí být v mnoha případech schopné získávat zkušenosti, či vůbec mít nějakou úroveň postavy (záleží na tipu hry), zatímco hlavní postavy nebude ovládat umělá inteligence, neboť je ovládá hráč.

Z tohoto zjištění můžu u dělat následující:

- Vytvořím si základní model postav, který bude obsahovat jen společné rysy pro všechny postavy
- Následně budu tento model využívat při tvoření libovolné skupiny postav jakožto základ.

Kvůli takovémuto přístupu mohu opakovaně používat model postavy a nemusím pracně vytvářet základní vlastnosti postav znova a znova.

3.3. Abstrakce

3.3.1. Popis

Abstrakce usnadňuje přemýšlení nad problémy nebo systémy. Abstrakce je zachycení struktury procesu, děje či problému. Při tom se snažíme nedávat pozornost nepodstatným nebo nepotřebným detailům. Vyhledáváme klíčové prvky v problému. [5] [7]

S použitím abstrakce volíme a vybíráme reprezentaci daných systémů. [7]

3.3.2. Příklady

Jako příklad z reálného života lze u vést mapu metra. Města, ve kterých se metra nacházejí bývají velmi složitými systémy. Znázornění takovýchto měst různými způsoby (obvykle mapami nebo obrázky) pomáhá různým uživatelům. Mapa metra je velmi propracovanou abstrakcí s dostatečnými informacemi, které cestovatel potřebuje k navigaci v podzemní síti bez zbytečného zatížení informacemi, jako je vzdálenost a přesná geografická poloha. Je to reprezentace, která obsahuje přesně informace nezbytné k naplánování trasy z jedné stanice do druhé – a nic víc. [5]

Dalším příkladem můžou být programy. Nejprve se učíme, co program dělá, pak se naučíme, jak programovat, pak možná o tom, jak jsou reprezentována data, pak o tom, jak načíst a spustit cyklus CPU a pak, ale pouze pokud nás opravdu zajímá elektronika, se naučíme, jak funguje, fyzika hardwaru. [9]

3.3.3. Příklad pro vývoj her

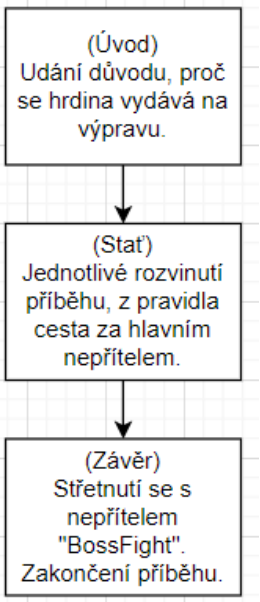
Skvělým případem abstrakce při vývoji her je často opomíjené téma, a sice tvorba příběhu. Mnoho lidí se domnívá, že hry, zvláště ty arkádové, často žádný příběh ani nemají, opak je však pravdou. i hry, kde na první pohled příběh není přímo prezentován vývojáři se příběh nachází, neboť příběh vzniká při samotném hraní hry ať u ž jej implicitně má či nikoliv. u ž jen samotná existence závěru hry s hlavním nepřítelem na konci, či možnost růstu hráčova postupu na základě dřívějších kroků v hráči vytváří pocit, že si určitý příběh prožívá. Je přitom tedy důležité vytvořit příběhovou konzistenci.[10]

Základem příběhu je osnova, kterou se většina z nás naučila vytvořit na základní škole. Tu lze rozdělit do tří základních částí a sice úvod, stať a závěr.

V úvodu hry může být například takzvaná „cutscéna“, neboli videosekvence, u dávající základ příběhu. Ve stati hry se nachází převážná většina hry, při které hráč získává schopnosti, peníze, rozvijí příběh, a to vše jen pro závěr, ve kterém se u tká s hlavním nepřítelem a rozuzlí příběh.

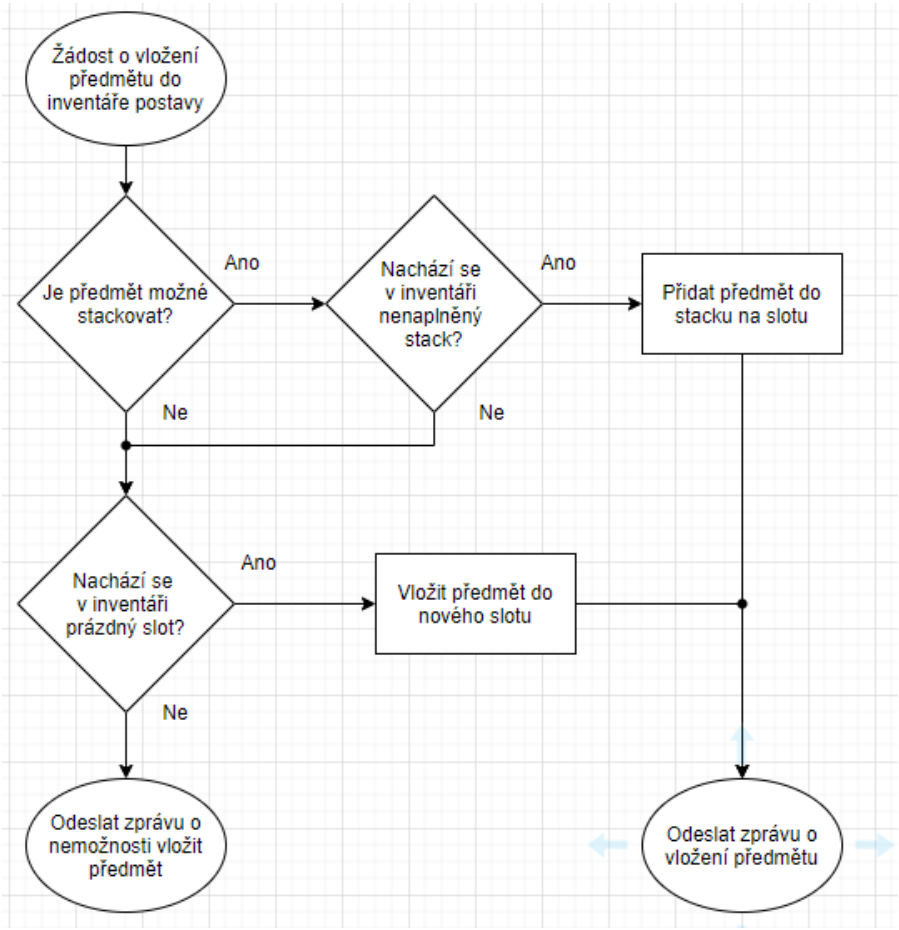
Některé hry mají možnost více možností, kam se příběh může rozvíjet a mohou mít klidně dvacet různých závěrů.

Ať u ž je příběh jakýkoliv, za každých okolností se řídí u rčitou osnovou, která je abstrakcí daného příběhu neboli odhlédnutím od detailů a zaměřením se na klíčové prvky.



OBRÁZEK 4 OSNOVA PŘÍBĚHU

Jiným případem abstrakce může být například schéma toho, jak bude fungovat systém přidávání předmětů do inventáře. V takovémto případě je dobré u dělat si základní návrh toho, jak by daný systém mohl fungovat. Musíme si pokládat otázky typu „Dají se předměty stackovat (sloučit do jednoho slotu)? Bude inventář mít konečný počet slotů? Je potřeba hlídat jeho naplněnost? Chci předměty rovnou po sebrání stackovat? Je to či ono klíčové pro fungování inventáře?“ a mnohé další.



OBRÁZEK 5 NÁVRH SYSTÉMU PŘIDÁVÁNÍ PŘEDMĚTŮ

Tímto přemýšlením, a následným řazením vytvořím obecnou strukturu problému neboli abstrakci, na což jí mohu realizovat prostřednictvím tvorby algoritmu.

3.4. Algoritmizace

3.4.1. Popis

Algoritmy jsou pokyny krok za krokem, jak něco u dělat, nebo soubor pravidel popisujících, jak něco funguje. Splní-li vykonavatel algoritmu (ať u ž se jedná o počítač či člověka) pokyny ve správném pořadí, měl by se stejnými vstupy pokaždé skončit se stejným výsledkem. [9] [11]

„Jedná se o pracovní postup, který má tyto povinné vlastnosti:

- ***Rezultativnost**, jenž znamená, že algoritmus vždy vydá nějaký výsledek.*
- ***Konečnost**. To znamená, že někdy skončí. Jinými slovy, skončí po konečném počtu provedených kroků. Programy, které nikdy neskončí nejsou algoritmy.*
- ***Elementárnost** neboli jednoduchost popisu. Algoritmus je popsán konečným počtem základních instrukcí. Tedy takových, o kterých je jasné, jak se provedou (a tedy neumožňují žádný osobitý výklad některého vykonavatele).*
- ***Determinovanost** neboli jednoznačnost. Postup práce je jasně daný a vždy závisí pouze na popisu algoritmu a na vstupu („pracovní materiál“, ať u ž jde o vejce, nebo o informace, tedy nějaká čísla). na průběh algoritmu nemá žádný vliv náhoda nebo svobodná vůle vykonavatele.“* [12]

3.4.2. Příklady

Příkladů algoritmů je mnoho. Může se jednat o popis cesty do školy, návod na postavení domu, popis práce při obdělávání pole, soubor tanečních kroků, či popis jízdy na kole. [5]

Jedním z dalších příkladů algoritmu může být například recept na pokrm. Při vaření pokrmu podle receptu nejprve zjistíme, zda máme všechny potřebné vstupy neboli suroviny, náčiní, či spotřebiče. Následně postupujeme podle předepsaného návodu v příslušném pořadí, až do chvíle, kdy není pokrm zhotoven.

Zhotovení pokrmu znamená, že z daného algoritmu vznikl nějaký výsledek, tedy, že algoritmus je rezolutivní.

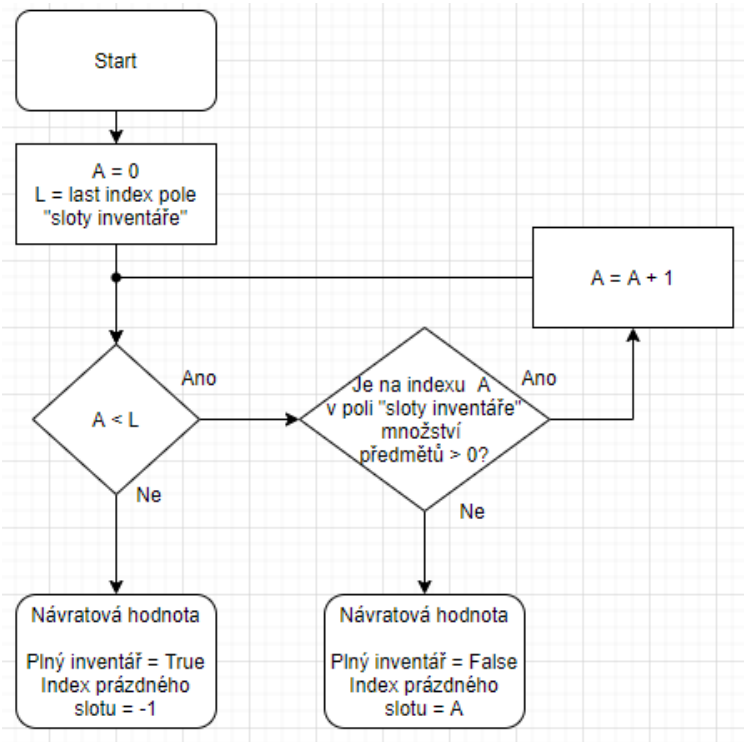
Po tom, co pokrm u vaříme, je algoritmus u končen, tedy má konec a je tedy konečný.

Při samotném vaření musím chápat všechny jednotlivé části receptu. Pokud jim nerozumím, může se snadno stát, že pokrm nebude zhotoven správně. Pokud však všemu rozumím, tak byl algoritmus dostatečně pochopitelný, jednoduchý a tedy elementární.

A nakonec je potřeba, aby byla z algoritmu přesně pochopena autorova myšlenka. Pokud autor napíše „vařte na mírném ohni“ a vykonavatel má elektriký sporák, tak se může snadno stát, že se pokrm nesprávně uvaří. Pokud by napsal, přidejte hrnek mléka a vykonavatelův hrnek by byl půllitrový, tak se taktéž pokrm nemusí u dělat správně. Pokud jsou však všechny instrukce napsány správně a není možné zpochybnit, co přesně autor daným výrokem myslel, pak je algoritmus jednoznačný, tedy determinovaný.

3.4.3. Příklad pro vývoj her

V kapitole pojednávající o abstrakci jsem představil návrh systému přidávání předmětů. Nyní zde předvedu, jak by mohl vypadat algoritmus vyhodnocující existenci prázdného slotu v inventáři.



OBRÁZEK 6 ALGORITMUS DETEKCE VOLNÉHO SLOTU

Samotné sloty inventáře nejsou ničím jiným než polem struktury, ve které jsou u loženy informace o předmětech (název, množství předmětu na slotu, …). na počátku algoritmu si nastavíme proměnou int a na 0 a proměnou int L na hodnotu posledního indexu v poli „sloty inventáře“. Následně procházíme polem pomocí cyklu. Při každém průchodu zjišťujeme, zda je struktura na daném indexu validní (například zjištěním hodnoty množství předmětů na slotu). Pokud na indexu validní předmět je, cykli se opakuje, pokud ne, algoritmus se u končí a vrátí hodnotu indexu prázdného slotu spolu se správou o existenci takového slotu. Pokud bude celé pole překontrolováno a prázdný slot nebyl nalezen, algoritmus vrátí zprávu o tom, že daný inventář je plný.

3.5. Generalizace

3.5.1. Popis

Jedná se o zobecnění postupů a činností. Při procesu generalizace identifikujeme vzory, podobnosti a využíváme tyto vlastnosti k možnosti znuvupoužitelnosti algoritmu. Je to způsob, jak rychle řešit nové problémy na základě předchozích zkušeností. Při generalizaci si kladu otázky typu: „Nepodobá se aktuální problém nějakému problému, který jsem již vyřešil?“ nebo „V čem je to jiné“. [6] [5]

„V objektově orientovaném přístupu se generalizace objevuje v souvislosti s pojmem dědičnosti, kdy třída předků objektu má všechny společné vlastnosti svých potomků. Potomci ale mohou mít vlastnosti změněné nebo nové a budou se od sebe vzájemně lišit.“ [8]

Jinými slovy, generalizace neboli zevšeobecňování nás vede k tvoření algoritmů, které můžeme v budoucnu využívat pro řešení různých situací. Lze u ní pozorovat následující vlastnosti:

- *„Identifikace vzorů a společných rysů.*
- *Přizpůsobení řešení nebo částí řešení tak, aby platily pro celou skupinu podobných problémů.*
- *Přenos nápadů a řešení z jedné oblasti problémů do druhé.“* [5]

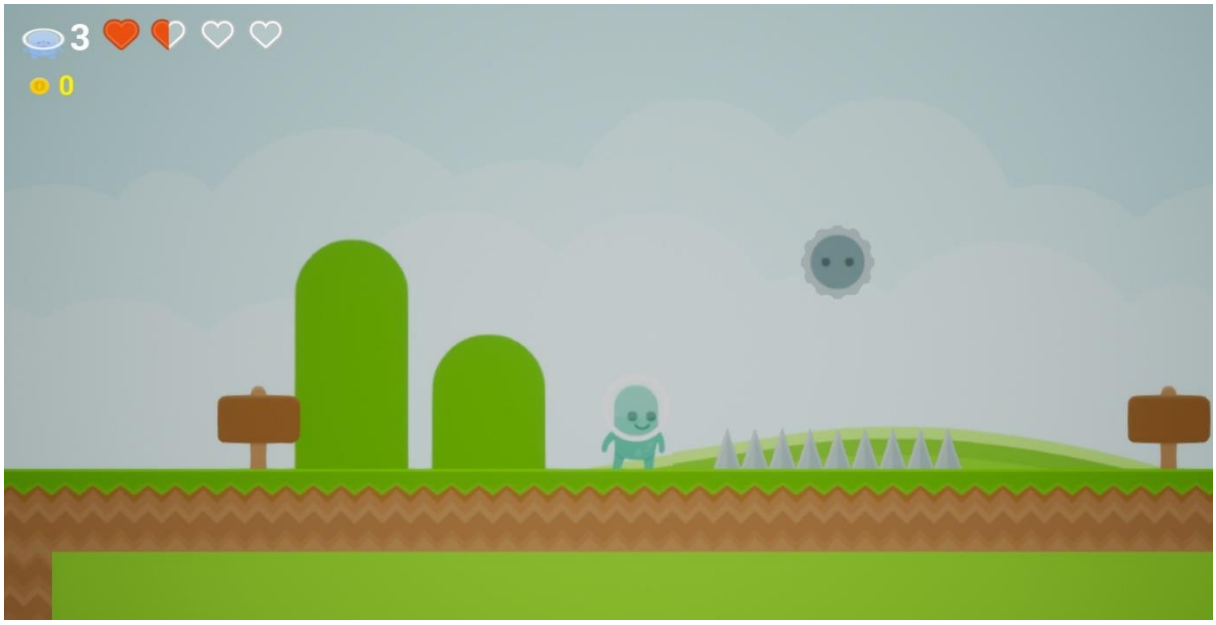
3.5.2. Příklady

Příkladem generalizace na základní škole může být například využívání tzv. „želví grafiky“. Při jejím používání žák kreslí různé tvary, jako jsou čtverec a trojúhelník. To vše za pomoci počítačového programu. Následně žák dostává úkoly typu: „Nakresli tvary dva“ či „Nakresli mnohoúhelníky o různých počtech stran“. Při práci s trojúhelníkem a čtvercem si všimnou, že existuje určitý vztah mezi počtem stran a počtem úhlů spolu s jejich velikostí v daném geometrickém útvaru. Následně žáci mohou navrhnout algoritmus, který jen na základě vstupních údajů dokáže nakreslit libovolný pravidelný mnohoúhelník. [5].

Dalším příkladem může být tvorba matematického algoritmu. Žáci mohou dostat za úkol navrhnout algoritmus kalkulačky. Při jejím návrhu mohou začít tvořit kalkulačku, která je ovládána pouze z konzole. Následně ji mohou vylepšit, aby byly vstupní hodnoty možné vygenerovat počítačem a následně mohou dostat za úkol, aby byla kalkulačka ovladatelná

pomocí grafického prostředí. Při každém z těchto kroků se kalkulačka stává použitelná pro čím dál více použitelná pro různé aplikace, tedy je zobecňována a probíhá u ní generalizace.

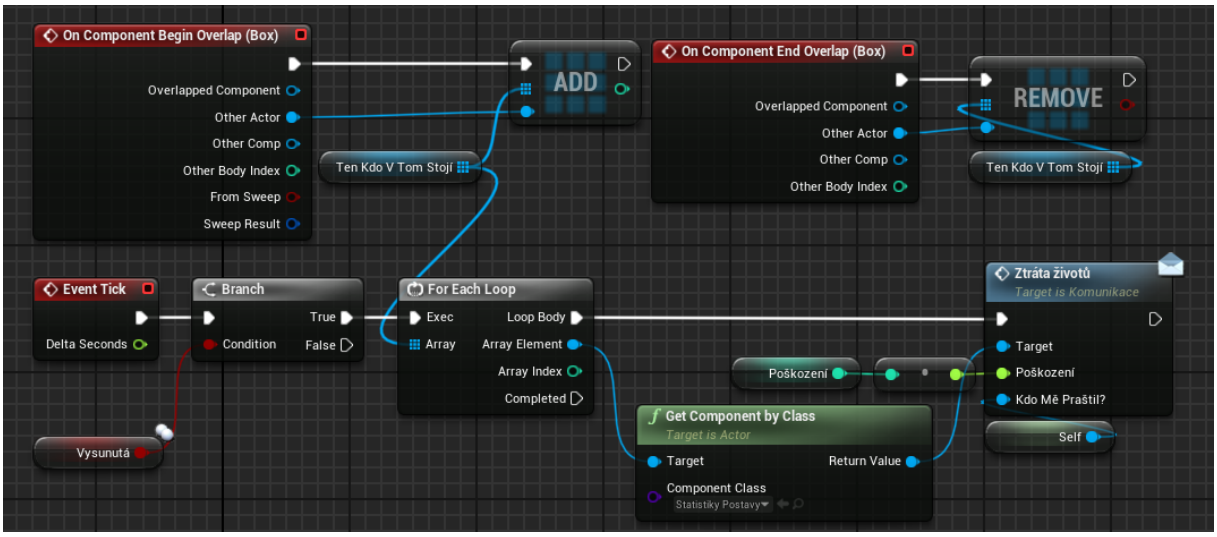
3.5.3. Příklad pro vývoj her



OBRÁZEK 7 U KÁZKA ŽÁKOVSKÉHO PROJEKTU

Na obrázku výše vidíte výřez obrazovky z projektu žáka z mého kroužku, který na tomto projektu pracuje i ve volném čase. Jeho hra je již velice dobře rozvinutá, takže počet životů, zdravý, ovládání postavy, sbírání zlata, oznamovací cedule, způsobování poškození a mnohé dalšího, co vidíte ve výřezu je již plně funkční.

Při popisu generalizace se zaměřím na určitou skupinu herců a sice těch, kteří při kontaktu s postavou způsobují postavě poškození (aktuálně bodce, ozubené kolečko, láva a nepřítel). Všichni totiž mají jednu společnou vlastnost a sice, že při kontaktu s postavou mají za určitých okolností způsobovat poškození.



OBRÁZEK 8 ALGORITMUS PRO DETEKCI ZPŮSOBENÍ POŠKOZENÍ

S pomocí generalizace byl vytvořen rodič, nadřazený všem těmto hercům, ve kterém byl vytvořen kód výše vyhodnocující, zda má být poškození udělováno, či nikoliv.

Událost „On Component Begin Overlap (Box)“ se zavolá ve chvíli, co nějaký herec způsobí kolizi s hercem působícím poškození. Jejím úkolem je zaznamenat všechny tyto herce do pole „Ten, kdo v tom stojí“.

Událost „On Component End Overlap (Box)“ se zavolá ve chvíli, co nějaký herec přestane být v kolizi s hercem působícím poškození. Jejím úkolem je odstranit všechny tyto herce z pole „Ten, kdo v tom stojí“.

Událost „Event Tick“ je volána pravidelně podle obnovovací frekvence obrazovky. po zavolání napřed zkontroluje, zda je herec v režimu, kdy způsobuje poškození kontrolou proměnné „Vysunutá“. Pokud je v takovém režimu, tak pravidelně odesílá informaci do všech herců z pole „Ten, kdo v tom stojí“ o tom, že jsou poškozeny pomocí cyklu a v něm volané události „Ztráta životů“. Při tom odesílá informaci o velikosti způsobeného poškození a odesílá odkaz na herce, který toto poškození způsobil (tudíž na sebe), pro další možnost interakce.

3.6. Optimalizace

3.6.1. Popis

Při optimalizování neboli vylepšování problému hledáme nejlepší možné řešení. To znamená, že se nespokojíme s prvním nalezeným řešením problému, ale snažíme se jich najít více a vybrat ten, který bude pro daný problém nejvhodnější. Ptáme se přitom na otázky: *„Který z postupů je lepší? Který je úspornější? V čem? Ve kterých případech a za jakých okolností? Jak?“* [13]

3.6.2. Příklady

Příkladem optimalizace může být snaha dopravit se z bodu a do bodu B. Při takovéto činnosti zvažuji mnoho okolností, jako je třeba volba dopravního prostředku. Půjdu pěšky, pojedu autem, autobusem, letadlem, vlakem, či na kole? Kromě dopravního prostředku volím také trasu, po které cestu realizuji. Pojedu poté či oné cestě? Pojedu přes dálnici? Pojedu rychlejší či pohodlnější trasou? Zvažováním všech možností optimalizuji cestu pro nejlepší možný výsledek. [13]

Při investování financí vybírám z velkého množství aktiv, z nichž každá má své výhody a nevýhody. Jaké očekávám průměrné roční zhodnocení? Budu investovat krátkodobě, či dlouhodobě? Budu investovat do akcií, komodit, nemovitostí, či kryptoměn? V jakém poměru? u akcií budu volit spíše růstové či dividendové společnosti? Z jakých zemí? Jaké jsou rizika? Atd. Při rozboru všech těchto aktiv optimalizuji nejlepší možné portfolio pro svůj styl investování.

3.6.3. Příklad pro vývoj her

Příklad, který zde u vedu, je ze stejného projektu stejného žáka jako ten, který jsem ukazoval v kapitole pojednávající o generalizaci. Z důvodu toho, že kód, který zde budu popisovat je integrovaný skrze tři různé třídy, nebudu zde z důvodu přílišné grafické složitosti zobrazovat výřez obrazovky.

Řešený problém byl následující. V postavě byla u ložena hodnota života a ta byla potřeba vypsat na obrazovku, aby hráč věděl, kolik právě jeho postava má života.

První verze tohoto výpisu vypadala tím způsobem, že ve třídě, která má na starosti zobrazování textbloků byl nastaven nekonečný cyklus, který neustále aktualizoval text textbloku na aktuální hodnotu života postavy. Jednalo se o jednoduchou aplikaci, která ovšem zbytečně moc vytěžovala systém. Bylo potřeba tedy tento systém optimalizovat.

Druhá verze fungovala na stejném principu s tím rozdílem, že kontrola probíhala každou sekundu. To mělo sice za následek zlepšení výkonu systému, avšak kvůli prodlevě zobrazení aktuálního života došlo k velikému diskomfortu pro hráče.

Při zkoušení různých možností nakonec vyhrála ta, ve které přímo při změně hodnoty zdraví byla vyslána hromadná událost, která spustila události ve třídách, v nichž bylo nastaveno sledování postavy. Tím jsme docílili toho, že vždy po změně zdraví sledované postavy se okamžitě změnila i hodnota zdraví zobrazující se na obrazovce. Tato aplikace měla navít tu výhodu, že krom vypisování hodnoty zdravý na obrazovku mohly být tou samou událostí volány libovolné události, které jsou potřeba spouštět po změně zdravý postavy.

3.7. Evaluace

3.7.1. Popis

Evaluace neboli vyhodnocování se zaměřuje na průběžné hodnocení daného řešení. Při tom se snažíme najít nejlepší možné řešení, za použití nejmenšího množství zdrojů. Je to postup, který se zaměřuje primárně na kladení otázek. Ptáme se při něm na otázky typu: *„Jsou správně? Jsou dost rychlé? Využívají zdroje ekonomicky? Jsou pro lidi snadné použít? Podporují vhodné zkušenosti?“* [13]

Mnohdy se setkáme se situací, kdy musíme pracovat s nějakými kompromisy, které musíme u činit (jen málokdy se naskytne jediné ideální řešení). Proto se ptáme: *„Je to dobré/pochopitelné/použitelné? Nemohli jsme to ještě rozložit na menší části? Je to kompletní? Řeší to opravdu všechny aspekty daného problému? Je to dost efektivní? Může to být ještě zlepšeno? Splňuje to všechna kritéria, která jsme si určili?“* [13]

3.7.2. Příklady

„Počítačová rozhraní jsou neustále vyvíjena tak, aby vyhovovala potřebám různých uživatelů. Pokud je například třeba zdravotnické zařízení k automatickému podávání léků pacientovi,

musí být programovatelné bezchybným, rychlým, jednoduchým a bezpečným způsobem. Řešení musí zajistit, že sestry budou schopny snadno nastavit dávku bez chyb a že to nebude frustrující pro pacienty ani zdravotní sestry, které budou řešení používat. V navrhovaném návrhu by měl být proveden kompromis mezi rychlostí zadávání čísel (efektivitou) a vyhýbáním se chybám (účinnost a použitelnost). Návrh by měl být posuzován na základě specifikace, kterou navrhli lékaři, regulátoři a odborníci na konstrukci zdravotnických prostředků (kritéria) a obecná pravidla dobrého designu (heuristika). Kritéria, heuristika a potřeby uživatelů umožňují systematicky a důsledně provádět rozhodnutí." [5]

Dalším příkladem muže být tvorba receptu na nějaký pokrm. Představte si, že na jste dovolený v indii ochutnali místní specialitu a vyžádali si recept s cílem, že si daný pokrm u vaříte doma. Když přijedete domu a chcete pokrm u vařit tak zjišťujete, že na u vaření daného pokrmu nemáte suroviny, které se prodávají v indii. Nyní musíte vyhodnotit, jak daný recept u pravit pro potřeby České republiky. Přidáte místo bambusu salát, nebo si raději necháte poslat bambus za dráž? Použijete tuto směs koření, či jinou?

3.7.3. Příklad pro vývoj her

Evaluaci můžeme pozorovat napříč celým vývojem her. na příkladu, který jsem uváděl v kapitole pojednávající o optimalizaci, ve kterém se řešil problém se zobrazením života.

Napřed byla neustále kontrolována hodnota zdraví, následně jen každou sekundu a až potom byla hodnota měněna pouze při změně života. Jednotlivý vývoj řešení by se nemohl vyvíjet bez použití evaluace neboli bez kladení otázek typu: Je to dobré? Využívá to zdroje ekonomicky? Až po zodpovězení na tyto otázky mohlo být vyvinuto lepší řešení pro danou situaci.

3.8. Shrnutí

3.8.1. Informatické myšlení a z čeho se skládá

Informatické myšlení je takový způsob myšlení, v němž uplatňujeme jak analytické, tak algoritmické přístupy k řešení problému. Rozvijí mysl tak že je následně schopna řešit složité i otevřené problémy za pomoci řady dovedností a technik. Běžná aplikace je psaní programů. [A14]

Mezi složky a postupy při využívání informatického myšlení se nejčastěji řadí: [13]

- Dekompozice (rozklad): schopnost rozdělit velký problém na menší řešitelné části
- Rozpoznávání vzorů: rozpoznávání vzorů a opakujících se částí
- Abstrakce: hledání obecné struktury problému a odhlédnutí od nepodstatného
- Návrh algoritmu: vývoj pokynů krok za krokem pro řešení tohoto a podobných problémů
- Generalizace: zobecnění algoritmu pro jeho budoucí potenciální využití a hledání podobných problému v předchozích řešení
- Optimalizace: schopnost a ochota nalézat nejlepší řešení
- Evaluace: schopnost přemýšlet z hlediska hodnocení

3.8.2. Důvody proč umět myslet jako informatik

„Všechny zmíněné dovednosti jsou podpořeny předpoklady a postoji, které jsou taktéž nezbytnou součástí informatického myšlení:

- ***Sebejistota tváří v tvář složitosti*** *(neuteču, když nevím, nenechám se odradit)*
- ***Vytrvalost při hledání řešení obtížných problémů*** *(nezdary a slepé uličky vnímám jako něco, co mě posouvá dopředu směrem k cíli)*
- ***Umět o problémech mluvit*** *(jsem schopen vidět, co je skutečným problémem, a říci to ostatním)*
- ***Tolerování nejednoznačnosti*** *(přijímám, že je třeba objevovat)*
- ***Schopnost pracovat s problémy s otevřeným koncem*** *(nevadí mi, že nevidím jejich řešení)*
- ***Schopnost dorozumět se a spolupracovat s ostatními při dosahování společného cíle*** *(umím naslouchat, ale také prosadit svoje, umím dát konstruktivní kritiku, jsem tolerantní k rozdílným pohledům, vnímám ostatní jako hodnotu)“* [13]

3.8.3. Složky informatického myšlení rozvíjené programováním

Z předchozích kapitol lze říct, že programováním se víceméně rozvíjejí všechny složky informatického myšlení, avšak některé více než ostatní. Pro potřeby této práce a efektivního vyučování je tedy potřeba vybrat ty ze složek, které jsou rozvíjeny nejvíce a nejefektivněji.

- Algoritmizace – s velkou mírou jistoty se jedná o složku, která je programováním rozvíjena nejvíce, neboť celé programování je v základu skládání jednotlivých kroků za sebe pro splnění daného cíle. Při tom jsou většinou splněny i jednotlivé podmínky pro definování algoritmu, jako je například elementárnost, či jednoduchost popisu.
- Dekompozice – Při samotném vývoji jsou většinou řešeny úkoly, které jsou komplexní. Při řešení těchto úkolů je potřeba problém rozdělit na menší části, které jsou již řešitelné.

- Vzory a sekvence – Složka informatického myšlení rozvíjená především při pracováním s cykly. Při využívání cyklů je velmi podstatné identifikovat daný vzor, aby na něj bylo možné efektivně aplikovat opakování.

4. Analýza úloh z pohledu vybraných složek Informatického myšlení

Samotná analýza úloh proběhla z pohledu tří složek informatického myšlení, a to sice algoritmizace, dekompozice a vzorů a sekvencí.

4.1. Algoritmizace

Při osvojování složky informatického myšlení algoritmizace se obecně využívá několik možných postupů.

Některé postupy kladou důraz na to, aby žáci nejdřív věděli, jak funguje hardware počítače a až poté se postupně u čili algoritmizaci (Hardware-first), jiné postupy využívají co možná nejširší záběr na informatiku, a až poté se žáci učí samotné algoritmizaci (Breadth-first). [14]

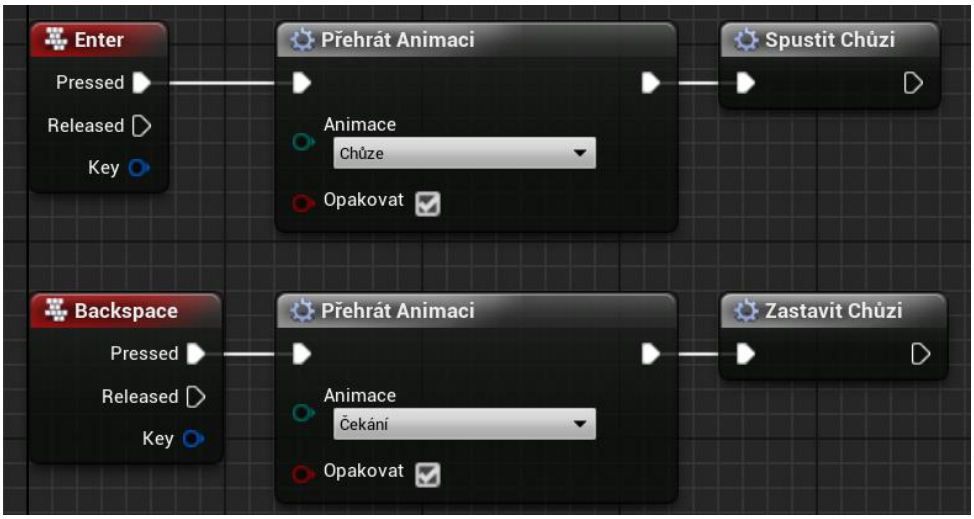
Postup, kterého využívá zkoumaná sada úloh se však liší v tom, že žáky vůbec nezatěžuje potřebou rozumět hardwaru počítače, či všeobecné informace, ale rovnou se zaměřuje na tvorbu algoritmů v objektovém prostředí. Z toho důvodu lze postup výuky algoritmizace označit za kombinaci modelů Imperative-first a Objects-first. Tyto metody rovnou seznamují studenty se základními programovými konstrukcemi, jako je opakování, podmínka, proměnná, přičemž úroveň objektivního programování je u Objects-first okamžitě od počátku výuky, zatímco u Imperative-first je přidána až po všech základních konstrukcích. [14]

Samotná sada úloh z počátku, učí pouze základní programové konstrukce, avšak od samotného počátku je v objektovém prostředí (postava v mapě). K samotné práci s objekty se kurz přesune až v části Volané události (drobně i v části vstupy a výstupy). Z toho důvodu kombinuje oba postupy.

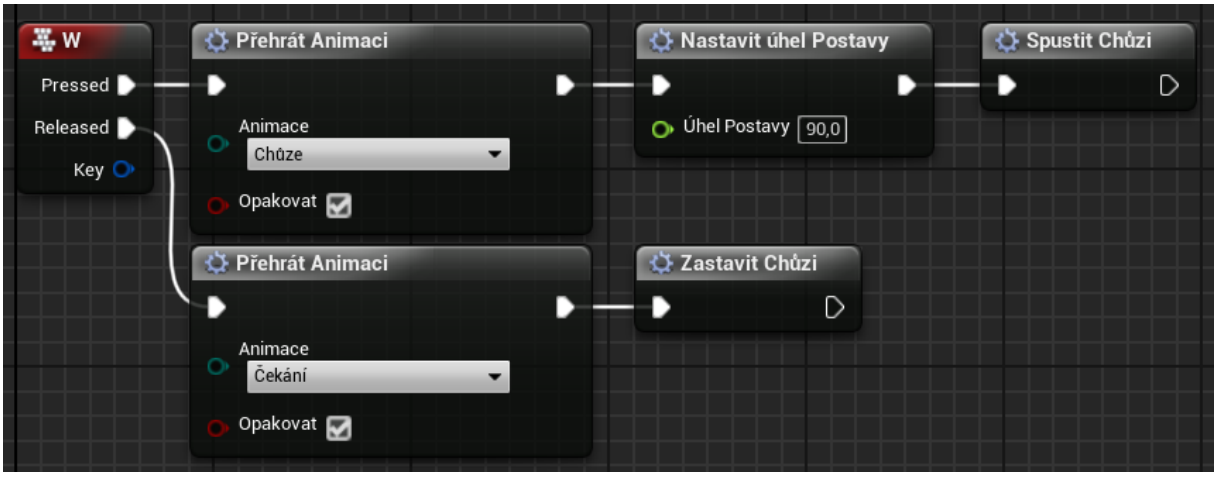
4.1.1. Obsah v úlohách

V úlohách je osvojování si algoritmizace téměř všudypřítomné. Většina úloh je koncipována tak, že žáci skládají jednotlivé bloky za sebe tak, aby se s určitostí vykonala nějaká událost, který je po daném kódu požadována v určitém čase. (Jsou obsaženy všechny části algoritmu: rezultativnost, konečnost, elementárnost, determinovanost).

Příkladem může být většina úloh, pro příklad mohu poukázat na úlohu ze 4. části 2 lekce, ve které je úkolem žáků různými způsoby rozpohybovat kostlivce za pomoci klávesnice.



OBRÁZEK 9 SCÉNÁŘE PRO START A KONEC CHŮZE



OBRÁZEK 10 SCÉNÁŘ PRO OVLÁDÁNÍ POSTAVY V JEDNOM SMĚRU

Závěrem lze tedy konstatovat, že sada úloh má potenciál k využití pro zlepšení v oblasti algoritmizace.

4.2. Dekompozice

Dekompozice nám může umožnit vyřešit na první pohled neřešitelný úkol. Rozložením problému na menší uchopitelné části nám může poskytnout přehled o tom, jak bychom měli postupovat při řešení určitého problému. Z toho je patrné, že při výuce žáků bychom se měli zaměřit na rozdělení problému na menší části, které jsou snáze realizovatelné. [15]

4.2.1. Obsah v úlohách

V samotných úlohách se s dekompozicí pracuje pouze ojediněle. Většina úloh je koncipována jednoduchým zadáním, které je složeno z několika málo kroků, což nevede žáky k potřebě

dekomponovat. Nejčastěji se dekompozice využívá u konce sady úloh v části Volané události a v části Proměnné. Například úplně poslední úloha je postavena tak, že je žákům pouze řečen požadovaný výsledek relativně komplexního kódu.

- Zajisti, aby se pokaždé, co Honza sebere nějaký předmět, napsal do chatu celou větou počet předmětů, jež Honza právě nese
- Zajisti, aby byl tento počet správný i potom, co Honza předá předměty Klaudii
- Zajisti, aby po předání předmětů Klaudii chat celou větou napsal, kolik předmětů Klaudie nese. Otestuj i případ, ve kterém Honza donese Klaudii předměty dvakrát.

OBRÁZEK 11 ZÁVĚREČNÝ ÚKOL

Z toho důvodu lze konstatovat, že sada úloh má částečný potenciál ke zlepšení schopnosti žáků dekomponovat.

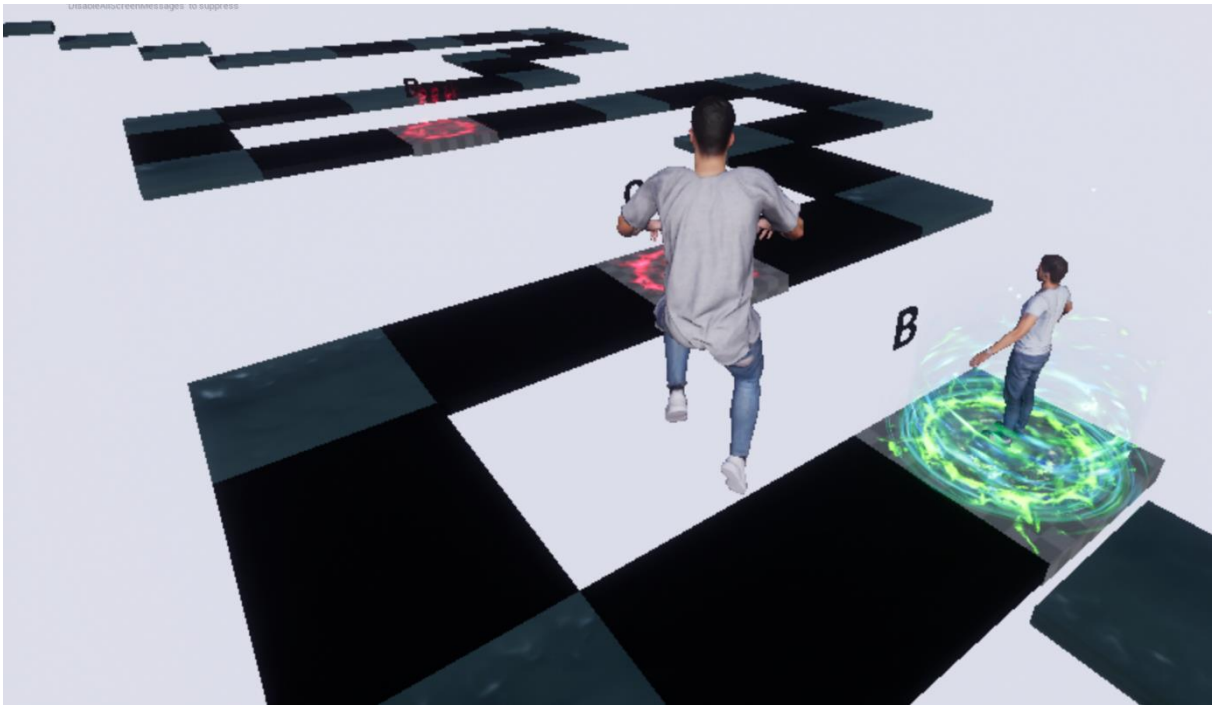
4.3. Vzory a sekvence

V programování se vzory a sekvence nejlépe vyučují v rámci tématu věnující se cyklům, z pohledu zkoumané sady úloh tedy v části Opakování bloků. [7]

Další oblast, ve které lze sledovat určitý vzor je část Vlastní bloky. Při tvorbě vlastní funkce (vlastního bloku) je potřeba znát určitý vzor, který požadujeme, aby funkce vykonávala. Může to být například vzor budovy s pozemkem, kterou mají žáci za úkol vytvořit ve 3. části 2. lekce.

OBRÁZEK 12 PŘÍKLAD DOMU S POZEMKEM

Identifikaci vzorů lze najít i ve 2. části 2. lekce, kde mají žáci identifikovat určitý vzor trasy, po které mají provést svou postavu za pomoci cyklů.



OBRÁZEK 13 U KÁZKA ÚROVNĚ LABYRINT

Z tohoto důvodu lze konstatovat, že práce obsahuje prvky vzorů a sekvencí a má tedy potenciál pro jejich rozvoj.

5. Akční výzkum

V současné době je akční výzkum považován za účinný způsob sebevzdělávání a nástroj změny v sociální realitě. Využívaný je v medicíně, v managementu, v sociálních vědách, ale i v pedagogice – zde v posledních třiceti letech zaznamenal mohutný rozvoj. Slouží jako nástroj profesního vzdělávání učitelů, rozvoje základního pilíře vzdělání (kurikula) a zlepšování dosavadní praxe. V české pedagogice je akční výzkum opomíjen, zatímco v zahraničí je mu věnováno velké pozornosti. [16]

Zakladatelem akčního výzkumu je americký sociální psycholog narozený v Německu Kurt Lewin. Právě Lewin jako první použil termín akční výzkum ve svém článku Action Research and Minority Problems (1946), kde kritizuje dosavadní přístup ke zkoumání organizací a skupinové dynamice. Se svým kolegou Frankem Baldauem začali pracovat na metodě mající potenciál reálně ovlivnit situaci v organizaci. Poté založili za účelem studia organizací a skupinové dynamiky výzkumné centrum (Research Center for Group Dynamics) při Massachusettském technologickém institutu. [17]

Akční výzkum je Lewinem vysvětlován jako spirálovitý proces. Ve chvíli zmapování problému je navrženo možné řešení, které je následně zrealizováno a sleduje se jeho úspěšnost. Cyklus se opakuje poté, co dojde k vyhodnocení a reflexi. Hlavním cílem je změnit dosavadní praxi, zásah do reality a zlepšení situace v komunitě, a to při dodržení vysokých standardů validity, reliability a vědecké přísnosti. [17]

Akční výzkum začal být využíván prakticky v oblasti vzdělávání, komunitního plánování a podnikového rozvoje, rozvíjel se také jako teoretická disciplína, kterou se zajímali sociology, antropology, pedagogy, psychology a další výzkumníky. [18]

5.1. Co je to akční výzkum?

Akční výzkum je zpravidla řazen do aplikovaného výzkumu v sociálních vědách. Přístup usiluje o změnu, není hodnotově neutrální a má politické pozadí (Hendl, 2005). Považujeme ho za významný prostředek inovací v sociální realitě. Je funkční především při transformačních procesech. Spojuje akci s výzkumem a vytváří most mezi skupinami účastníků (Walterová, 1997). Akční výzkum – tento termín však může být velmi matoucí, zastřešuje totiž celou rodinu přístupů vycházejících z různých politických, filozofických a metodologických zdrojů. [18]

Není tudíž možné obsáhnout všechny jeho podoby v jedné definici. Lze ovšem popsat na čem je založena metodologie akčního výzkumu a jaká jsou jeho východiska, nalézat společné rysy jednotlivých přístupů a soustředit se na specifika akčního výzkumu. Základ každého akčního výzkumu tvoří tři složky, které mu dávají konkrétní podobu – výzkum, akce a formativní složka. [18]

- Výzkum – v souvislosti s akčním výzkumem je vnímán jako reflektivní, systematický, kontrolovaný a kritický projekt, jehož cílem je prozkoumání jednotlivých aspektů reality související s řešeným problémem. Cílem je u tvořit si u celený a detailní obraz situace.
- Akce – jedná se především o následnou intervenci a implementaci navržených opatření.
- Formativní složka – účastníci jsou po celou dobu výzkumu konfrontováni s okolím a podstupují trvající sebereflexi. Mění se vztahy ve skupině, ale mění se také postoje a hodnoty samotných účastníků.

Tyto tři složky (akční, výzkumná a formativní) jsou charakterem akčního výzkumu.

5.2. Akční výzkum v učitelské praxi

Habermas kladl důraz na důležitost lidské zkušenosti při vytváření nového poznání. Jen krůček odtud vedl k přesvědčení, že učitelé by měli využívat své vlastní zkušenosti ke zlepšení výuky a nespoléhat jen na doporučení, která jim akademický výzkum nabízí. Akční výzkum učitelů vychází jako metoda pro zhodnocení a zlepšení praxe učitelů. Může jít o aktivitu jedince (učitele), ale také o komunitní projekt, jehož cílem je řešení problémů ve škole. V současnosti se akční výzkum v pedagogice těší velkému zájmu a věnuje se mu velké množství autorů. Akční výzkum je využíván praktiky, ale těší se také zájmu akademiků. [18]

6. Ověřování úloh

Samotné ověřování úloh probíhalo na Základní škole v Choustníku v rámci kroužku programování pod mým vedením. Věková skupina, která se účastnila testování byla od 10 do 16 let a zároveň byly zastoupeny všechny třídy druhého stupně. Mezi účastníky byli jak žáci základní školy Choustník, tak i dva žáci z jiných škol.

Ověřování probíhalo zhruba dva roky poté, co byla vytvořena sada úloh při testování s první skupinou žáků. Výsledky tohoto testování jsou tedy výsledky z pozorování druhé skupiny.

Ověřování probíhalo s přestávkami z důvodu epidemie Covid-19 a jiných událostí (vánoční prázdniny, pobyt na horách při kurzu lyžování, …) od pátku 12.11.2021 do neděle 3.4.2022. Setkání probíhala maximálně jednou týdně, a to buď v pátek, nebo v neděli (záleželo na tom, zda žáci mohli přijít v pátek).

Typický průběh jednoho setkání vypadal tak, že se hodinu a deset minut pracovalo na projektech (počítače a projekty byly již před příchodem žáků zapnuté a připravené).

OBRÁZEK 14 FOTOGRAFIE ŽÁKA PRACUJÍCÍ NA ÚLOZE VĚNUJÍCÍ SE CYKLŮM

Celkem kroužek navštěvovalo 15 žáků, avšak z důvodu velkých pauz v průběhu testování celý obsah kroužku završilo 9 žáků. Níže u vedená tabulka zobrazuje účast zkoumaných žáků na jednotlivých hodinách podle následujícího kritéria:

- Zaškrtnuté políčko: žák byl přítomen

- Nezaškrtnuté políčko: žák chyběl

Dále je barevně rozlišeno, kteří žáci patřili, do jaké skupiny, celkem tedy byly 4 skupiny, které absolvovaly celou sadu úloh.

Účast																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
[1] Erik I	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐	☑	☐
[2] Erik II	☑	☑	☐	☑	☑	☑	☑	☑	☐	☐	☑	☑	☑	☑	☐	☑
[2] Ondřej	☑	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[3] Vojtěch	☑	☑	☑	☐	☑	☑	☐	☑	☑	☑	☐	☑	☑	☑	☐	☑
[3] Aneta	☑	☐	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[4] Eliška	☑	☑	☐	☑	☐	☐	☑	☑	☑	☑	☑	☐	☑	☐	☐	☐
[4] Lucie	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑	☑	☑
[4] Nikola	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

TABULKA 1 ÚČAST ŽÁKŮ V JEDNOTLIVÝCH HODINÁCH

6.1. Ověřování cílů výuky

Použil jsem metodu akčního výzkumu, kdy jsem přímo v roli učitele pozoroval, jak žáci pracují, co konkrétně jim jde a co nejde. u každé odučené hodiny jsem také analyzoval žákovské práce. V průběhu výuky jsem provádět rozhovory nebo dotazování žáků. Tyto tři podstatné prvky hodnocení naplnění vzdělávacích cílů nyní rozvedu:

- Průběžné sledování aplikování poznatků z minulých lekcích na těch aktuálních

  Pro příklad: Pokud cílem výuky předchozí lekce mělo být to, že žák rozumí důvodu pořadí bloků a díky této dovednosti dokáže plnit jednoduché úkoly, tak v následujících lekcích bylo pozorováno, zda žáci danou dovednost ovládají (pokud žák sestaví scénář z důvodu učení se opakování, pak bylo pozorováno, zda jsou bloky zapojeny ve správném pořadí).

- Analýza odpovědí žáků v pracovních listech

  Pracovní listy pro žáky obsahují otázky, na které žák v průběhu jednotlivých lekcí odpovídá. Část těchto otázek je určena k ověření, zda žák pochopil danou problematiku, či nikoliv. Analýzou těchto otázek bylo možné objasnit, zda žák zvládne popsat nějaký blok, či princip a potažmo zda ovládá prvek daného cíle výuky.

- Průběžné rozhovory a dotazy

  V průběhu výuky byli žáci často dotazováni na důvody, proč se daný scénář chová tak, či onak. Ku příkladu u opakování byli žáci často dotazováni na pořadí, ve kterém se bloky aktivují z důvodů procvičení a zjištění, zda žáci chápou, kdy je cyklus u končen a co po něm následuje.

O konkrétních cílech výuky pojednávám v následujících osmi kapitolách. Tabulky, které ukazují naplnění, či nenaplnění cíle výuky vycházejí z absenční tabulky z předchozí kapitoly a mají následující kritéria:

- Zaškrtnuté políčko: žák byl přítomen
- Nezaškrtnuté políčko: žák chyběl
- Černé políčko: v daný moment nezkoumáno
- Červené políčko: cíl výuky v daný moment nebyl naplněn
- Žluté políčko: cíl výuky byl v daný moment naplněn částečně
- Zelené políčko: cíl výuky byl naplněn

6.1.1. Část sestavování scénáře

Žák dokáže spustit projekt vytvořený v UE4

Jedná se o jednu z nejzákladnějších schopností, která byla potřeba aby si žáci osvojili. Důležitá byla především v dobách distanční výuky kdy jediným způsobem, jak jsem mohl pokračovat v ověřování sady úloh byla práce žáků z domova tedy bylo nutné, aby žáci byli schopni svůj projekt bez jakýchkoliv problémů sami spustit doma.

Ověřování této dovednosti bylo prosté. Vždy na úvodu kroužku jsem sledoval, zda žáci mají či nemají problém se zapnutím projektu. Samozřejmě vzhledem k tomu že žáci pracovali ve dvojicích bylo nutné ověřit oba členy skupiny nebo většinou se stávalo že projekt zapínal jenom jeden a druhý mezitím dělal něco jiného. Z toho důvodu jsem ještě jednou ke konci hodiny poprosil vždy žáka, který projekt nezapínal, aby se jí pokusil zapnout.

Většina žáků byla schopná hned po první hodině projekt opětovně spustit zbylí žáci toho byli schopni hned v následujících hodinách.

Celkově tedy lze konstatovat že tento cíl výuky byl u všech žáků naplněn.

Žák dokáže spustit projekt vytvořený v UE4																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
[1] Erik I	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐	☑	☐
[2] Erik II	☑	☑	☐	☑	☑	☑	☑	☑	☐	☐	☑	☑	☑	☑	☐	☑
[2] Ondřej	☑	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[3] Vojtěch	☑	☑	☑	☐	☑	☑	☐	☑	☑	☑	☐	☑	☑	☑	☐	☑
[3] Aneta	☑	☐	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[4] Eliška	☑	☑	☐	☑	☐	☐	☑	☑	☑	☑	☑	☐	☑	☐	☐	☐
[4] Lucie	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑	☑	☑
[4] Nikola	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

TABULKA 2 ŽÁK DOKÁŽE SPUSTIT PROJEKT VYTVOŘENÝ V UE4

Žák dokáže sestavit scénář, jímž splní zadaný úkol

Tento cíl výuky je poměrně obecný proto bych její nyní chtěl upřesnit. Je jím myšleno, že žák rozumí důvodu pořadí bloku. Chápe, že blok, který je napojen dříve, se vykoná jako první, a až následně se vykoná blok, který je napojen dále. Kvůli tomuto poznatku dokáže žák plnit jednoduché úkoly.

V lekci 1-1 žáci programovali generování cesty, po které mohly projít s postavou, jenž ovládali. Žáci měli ze začátku v této části problémy s pochopením toho, proč je zde u místěno více událostí, což splnění tohoto cíle výuky mírně zkomplikovalo, avšak nejednalo se o žádný závažný problém, neboť od vytváření třetí události již žáci důvod více událostí pochopili.

V lekci 1-2 žáci programovali animaci a pohyb postavy. Ke klasickému skládání bloků do série, přičemž se všechny bloky okamžitě vykonají, byla přidána další vlastnost, a to sice čas po který se daný blok vykonává. Nějakou dobu trvalo, než si žáci tuto vlastnost osvojili, avšak lze říct, že všichni žáci nakonec práci s časem pochopili.

V lekci 1-3 žáci pracovali v mapě určené pro první část a ovládali postavičku z druhé části. Tato část sloužila především pro u celený si poznatků o sestavování scénáře.

Ověření schopností žáků sestavit scénář na základě daného úkolu bylo možné realizovat způsobem sledování žáků při vytváření scénáře v následujících úlohách. Pokud žák tápal nad tím, jak u dělat například cykli, ale věděl, v jakém pořadí mají bloky být pak jsem vyhodnotil, že byl cíl výuky naplněn.

Pokud žák zapojil scénář ve špatném pořadí, ale dokázal na svou chybu přijít z největší části sám pak jsem vyhodnotil, že danou hodinu byl cíl dílky splněn částečně.

Pokud žák nedokázal chybu najít sám a netušil proč mu scénář nedělá to co potřebuje vyhodnotil jsem, že pro danou hodinu nebyl cíl výuky splněn.

Ze sesbíraných dat mohu konstatovat, že většina žáků byla po lekcích určených k naučení základní práce se scénářem schopna bezproblémově pracovat tudíž tento cíl výuky považuji za splněný.

Žák dokáže sestavit scénář, jímž splní zadaný úkol >> Žák rozumí důvodu pořadí bloků, a dokáže plnit jednoduché úkoly																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✓	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✓	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABULKA 3 ŽÁK DOKÁŽE SESTAVIT SCÉNÁŘ, JÍMŽ SPLNÍ ZADANÝ ÚKOL

Žák dokáže scénář upravovat mazáním bloků, či změnou jejich pořadí

Jedná se o dovednost, která mimo jiné podmiňuje i splnění předchozího cíle výuky. Jedná se o schopnost základní manipulace s jednotlivými bloky tak, aby z nich bylo možné sestavit scénář. Mimo jiné v sobě tento bod obsahuje i vytváření takzvaných "nodes", které jsou užitečné pro zlepšení přehlednosti kódu, neboť kvůli nim je možné vizuálně měnit dráhu signálu.

OBRÁZEK 15 NODES

Ověřování osvojení této dovednosti jsem prováděl tak, že jsem přišel k žákovi a sledoval jsem, jak vytváří scénář pro aktuální úlohu. Většinou jsem čekal až do momentu, kdy žák přidal blok, který do scénáře nepatřil nebo jej dal na špatné místo. Od této chvíle jsem pozoroval, zda žák zvládne sám objevit tuto chybu a blok smazat či přesunout. Pokud na to nezvládl přijít sám stačilo mi, že po instrukci, aby nějaký blok smazal či přesunul, toto zadání zvládl.

Ze shromážděných dat mohu konstatovat, že největší problém měli žáci právě s vytvářením "nodes". Z toho důvodu má taky většina žáků při prvním testování oranžovou barvu. **Co se týče samotné práce s bloky lze říct, že všichni žáci velice rychle pochopili, jak je efektivně mazat a měnit jejich pořadí čili tento cíl výuky považuji za splněný.**

Žák dokáže scénář upravovat mazáním bloků, či změnou jejich pořadí																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✓	✓	✓	✓	✓	✓	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✓	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✓	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABULKA 4 ŽÁK DOKÁŽE SCÉNÁŘ U PRAVOVAT MAZÁNÍM BLOKŮ, ČI ZMĚNOU JEJICH POŘADÍ

Žák dokáže spustit simulaci a ověřit správnost svého scénáře

Klíčová dovednost potřebná k tomu, aby žák byl schopen ověřit, zda to co naprogramoval funguje tak, jak si představoval, či nikoliv.

OBRÁZEK 16 TLAČÍTKA SIMULATION A PLAY

Během testování této schopnosti jsem narazil na drobnou komplikaci, a to sice tu, že v módu zobrazení mapy je tlačítko play jasně pochopitelné tím způsobem, že vedle něj není žádné jiné tlačítko, které by evokovalo myšlenku, že by mělo vykonávat podobnou funkci. Když však přejdeme do Blueprints, tak můžeme narazit na to, že vedle tlačítka play je ještě tlačítko simulation což žáky mátlo. Toto tlačítko je určeno primárně pro pokročilé zkoumání kódu, avšak tato funkce je příliš náročná pro potřeby základní školy.

Kromě tohoto drobného problému, který se vyřešil v první části lze konstatovat, že jsem nezaznamenal žádný případ, kdyby žáci nevěděli, jak spustit simulaci a tím si ověřit funkčnost svého kódu. s ohledem na to konstatuji že tento cíl výuky byl naplněn.

Žák dokáže spustit simulaci a ověřit správnost svého scénáře																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
[1] Erik I	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐	☑	☐
[2] Erik II	☑	☑	☐	☑	☑	☑	☑	☑	☐	☐	☑	☑	☑	☑	☐	☑
[2] Ondřej	☑	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[3] Vojtěch	☑	☑	☑	☐	☑	☑	☐	☑	☑	☑	☐	☑	☑	☑	☐	☑
[3] Aneta	☑	☐	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[4] Eliška	☑	☑	☐	☑	☐	☐	☑	☑	☑	☑	☑	☐	☑	☐	☐	☐
[4] Lucie	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑	☑	☑
[4] Nikola	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

TABULKA 5 ŽÁK DOKÁŽE SPUSTIT SIMULACI A OVĚŘIT SPRÁVNOST SVÉHO SCÉNÁŘE

6.1.2. Část opakování bloků

Žák ví, jak funguje opakování

Jedná se o nejpodstatnější cíl, který tato lekce obsahuje. Splněním tohoto cíle se rozumí, že žák chápe princip opakování. Neboli, že je možné bloky zacyklit tak, aby stejná část kódu byla provedena vícekrát a má představu o tom, jak je možné to u dělat.

Lekce 2-1 slouží k tomu, aby žák plynule přešel z jednoduché tvorby scénáře, až k prvnímu použití opakování. Mírná komplikace nastala s tím, že žáci často zapomínali kód zacyklit bílou čarou tak, aby mohl signál procházet dokola. To mělo za následek, že se program zastavil po prvním cyklu a pak již nic nevykonal. Tuto chybu však v průběhu zkoušení většina žáků přestala dělat a když u ž ji u dělali tak i velice rychle identifikovali a opravili.

Lekce 2-2 se zaměřuje především na procvičení aplikace opakování v jiné situaci. V ní žáci programují postavičku tak, aby prošla určitou dráhou s tím, že měli využívat opakování. Během testování této části se u kázalo, že žáky ne zcela dobře vede k tomu osvojit si opakování, neboť většina žáků během této části opakování buď nepoužila vůbec, nebo jej použila až po mém vyzvání (až na jednu výjimku). Kvůli tomu mohu říct, že druhá část potřebuje úpravy.

V lekci 2-3 žáci předělávat kód který, vytvářely tak aby obsahoval méně bloků. Toho měli docílit s použitím opakování. V této části se u kázalo, že žáci rozumějí principu opakování a dokážou jej víceméně aplikovat. Někteří žáci byly výrazně rychlejší jiným bylo potřeba dopomoct, avšak všichni tuto část zvládli.

Během ověřování tohoto cíle výuky jsem bral v potaz, jak praktické úlohy, kdy žáci přímo vytvářely svůj program taky odpovědi přímo žákovských listech.

Pokud žák nedokázal ani vysvětlit, jak opakování funguje, ani prakticky opakování aplikovat tak byl označen na dané hodině červeně. Pokud žák dokázal buďto odpovědět, jak opakování funguje, nebo prakticky opakování využít v programu, byl označen žlutě. Pokud žák zvládl obojí neboli jak vysvětlit opakování i jej aplikovat byl označen zeleně.

Ačkoliv lekce 2-2 nevyšla dle očekávání, tak kvůli třetí části mohu konstatovat, že žáci byli schopni pochopit, jak opakování v programování funguje. Kvůli tomu mohu u soudit, že tento daný cíl výuky je splněn.

Žák ví, jak funguje opakování																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✓	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✓	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABULKA 6 ŽÁK VÍ, JAK FUNGUJE OPAKOVÁNÍ

Žák pozná, kdy je vhodné použít opakování

Touto větou je myšleno, že žák dokáže identifikovat opakující se části kódu, nebo opakující se činnosti a s tímto vědomím vhodně aplikovat opakování. Zároveň žák dokáže v již hotovém scénáři identifikovat sekvenci, která se opakuje a tu přetvořit s použitím opakování.

Při testování jsem narazil na problém lekce 2-2, který popisuji v předchozí kapitole, kvůli kterému si žáci plně neosvojili schopnost rozpoznat, kdy je při řešení problému lepší použít opakování (až na výjimky).

Kvůli třetí lekci však mohu říct, že žáci jsou schopni scénář zjednodušit, pokud se na něj zaměří. Zároveň v následujících částech žáci byly schopni tvořit jednoduchá opakování (např ve druhé části 3. lekce).

Z těchto dat nelze říct, že si žáci plně osvojili schopnost rozpoznat, zda opakování intuitivně využít, či nikoliv. na druhou stranu žáci jsou schopni po absolvování těchto lekcí opakování v triviálních případech využívat. Z toho důvodu jsou data k této otázce vyhodnocena ve většině případů jakožto neprůkazné.

Žák pozná, kdy je vhodné použít opakování																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✗	✗	✓	✓	✓	☐	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✗	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✗	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✗	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✗	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

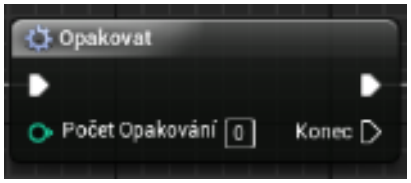
TABULKA 7 ŽÁK POZNÁ, KDY JE VHODNÉ POUŽÍT OPAKOVÁNÍ

Žák dokáže do scénáře, vložit opakování, pokud jej scénář neobsahuje

Touto dovedností je myšlena především „technická“ způsobilost žáků při využívání opakování. Klíčové vědomosti, které žáci musí ovládat je vědomí toho, že aby se mohlo aplikovat opakování, tak je potřeba kód zacyklit neboli hlavní propojovací čáru zapojit do scénáře zpět.

OBRÁZEK 17 ZAPOJENÍ SCÉNÁŘE DO CYKLU

Druhá podstatná dovednost je rozumět u bloku „Opakovat“ tomu, že změnou parametru „Počet opakování“ se mění počet opakovaných cyklů.



OBRÁZEK 18 BLOK OPAKOVAT

Třetí klíčovou dovedností je u rčení toho, kdy cyklus končí a co po něm následuje neboli pochopení funkce zobáčku „Konec“.

Při testování tohoto cíle výuky žáci z počátku občasně zapomínaly na všechny tyto dovednosti, avšak po několika vytvořených opakováních se žáci všem klíčovým dovednostem naučili. Kvůli tomu mohu konstatovat, že daný cíl výuky byl naplněn.

Žák dokáže do scénáře, vložit opakování, pokud jej scénář neobsahuje																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
[1] Erik I	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐	☑	☐
[2] Erik II	☑	☑	☐	☑	☑	☑	☑	☑	☐	☐	☑	☑	☑	☑	☐	☑
[2] Ondřej	☑	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[3] Vojtěch	☑	☑	☑	☐	☑	☑	☐	☑	☑	☑	☐	☑	☑	☑	☐	☑
[3] Aneta	☑	☐	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[4] Eliška	☑	☑	☐	☑	☐	☐	☑	☑	☑	☑	☑	☐	☑	☐	☐	☐
[4] Lucie	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑	☑	☑
[4] Nikola	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

TABULKA 8 ŽÁK DOKÁŽE DO SCÉNÁŘE, VLOŽIT OPAKOVÁNÍ, POKUD JEJ SCÉNÁŘ NEOBSAHUJE

6.1.3. Část vlastní bloky

Žák dokáže vytvořit a pojmenovat vlastní blok

Jedná se o technickou dovednost v UE4 při tvorbě vlastních bloků (v jazyku UE4 „Macros“). Tato dovednost je klíčová pro následující cíl výuky a zároveň žákovy poskytuje pohled na to, jak fungují bloky, se kterými doposud pracuje (pohled toho, že vlastně všechny bloky, se kterými procuje jsou též jen někým vytvořená makra, funkce bloky, …).

Lekce 3-1 navazuj na lekci 2-2 z předchozí části. Zde mají žáci za úkol tvořit dráhu pohybu pro postavu Honzu a následně z této cesty vytvořit blok. Tato lekce se věnuje primárně tomuto cíli výuky, neboť v UE4 je vytvoření vlastního a následná orientace mezi kódem v bloku a kódem ve zbytku scénáře poměrně náročné pro žáky základní školy. Z toho důvodu je poměrně monotónní.

Největší problém, se kterým žáci pracovali byl, při samotné tvorbě bloku vybrat se seznamu po zobrazeného po klinutí pravého tlačítka myši vhodnou akci, a to sice „Collapse to Macro“.

Pro mě překvapivě po úspěšném zvládnutí první lekce této části již žáci neměli problém s vytvářením vlastních bloků, ač jsem předpokládal, že postup jejich tvorby část žáků zapomene. Z tohoto důvodu hodnotím tento cíl výuky jakožto splněný.

Žák dokáže vytvořit a pojmenovat vlastní blok																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✓	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✓	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABULKA 9 ŽÁK DOKÁŽE VYTVOŘIT A POJMENOVAT VLASTNÍ BLOK

Žák dokáže se svými bloky pracovat ve scénáři

Daný cíl výuky znamená, že je žák schopen intuitivně poznat, kdy je vhodné vytvořit si svůj vlastní blok a efektivně s ním pracovat.

K tomuto účelu měla sloužit především lekce 3-2, která navazuje na lekci 2-1. V této lekci mají žáci za úkol vytvořit scénáře, které vytvoří dům se zahradou a tento pozemek následně u ložit jakožto blok. Cílem je, aby žáci získali nadhled k tomu, že blok je kombinace jiných bloků tvořící nějaký vyšší celek (pozemek, dráha pohybu, …).

Pokud bych měl hodnotit pouze schopnost pracovat s vlastními bloky podle instrukcí, mohl bych konstatovat, že byl cíl výuky splněn. Avšak v cíli výuky je obsažena i vize, že žák intuitivně pozná, kdy je vhodné vlastní blok vytvořit. Kvůli sledování následujících úloh, především pak těch v části Parametry je nutno konstatovat, že se tato vize u žáků nenaplnila.

Z výše u vedeného lze konstatovat, že cíl výuky byl splněn částečně. Žáci jsou schopni s vlastními bloky pracovat podle instrukcí, avšak ne intuitivně.

Žák dokáže se svými bloky pracovat ve scénáři																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
[1] Erik I	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐	☑	☐
[2] Erik II	☑	☑	☐	☑	☑	☑	☑	☑	☐	☐	☑	☑	☑	☑	☐	☑
[2] Ondřej	☑	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[3] Vojtěch	☑	☑	☑	☐	☑	☑	☐	☑	☑	☑	☐	☑	☑	☑	☐	☑
[3] Aneta	☑	☐	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[4] Eliška	☑	☑	☐	☑	☐	☐	☑	☑	☑	☑	☑	☐	☑	☐	☐	☐
[4] Lucie	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑	☑	☑
[4] Nikola	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

TABULKA 10 ŽÁK DOKÁŽE SE SVÝMI BLOKY PRACOVAT VE SCÉNÁŘI

6.1.4. Část vstupy a výstupy

Žák dokáže vytvářet události závislé na interakci s uživatelem

Tento cíl výuky je opět zaměřen spíše na technickou způsobilost žáka. Cíl výuky u dává požadavek na žáka ve smyslu schopnosti vytvářet samotné události závislé na základních vstupech při komunikaci s počítačem, a to sice vstupy z klávesnice a myši. Zároveň u dává i schopnost pracovat s výstupy, a to sice v podobě výpisu z „konsole“, nebo rozšíření výstupu v podobě propojení možnosti přímého ovládání funkcionalit scénářů, jako je například možnost ovládat postavu.

Lekce 4-1 je zaměřena na seznámení žáka se vstupy s použitím myši skrze tlačítka v obrazovce. Zároveň jde o první lekci, kdy žáci z důvodu konstrukce UE4 potřebují využívat proměnou odkazující na jiný objekt, než je plán, ve kterém tvoří, a to sice objekt Chat a objekty tlačítek. Z toho důvodu se lekce věnuje z velké části i této dovednosti.

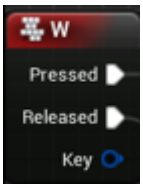
Lekce 4-2 se věnuje použití vstupů z klávesnice. Žáci v této lekci pracují na způsobu, jak ovládat postavu kostlivce s použitím klávesnice. Tato lekce byla pro žáky tedy jednodušší, neboť žáci pracovali s ovládáním postavy, což jim je důvěrně známo.

Samotné hodnocení splnění cíle výuky bylo závislé na dvou bodech:

1. Žáci jsou schopni vytvořit událost s pomocí „Event dispecheru“ (funkce UE4, která umožňuje vytvořit události, které jsou vlastní nějakému objektu, v tomto případě událost tlačítek po stisknutí myši).

OBRÁZEK 19 SEZNAM EVENT DISPECHERŮ TLAČÍTKA

2. Žáci jsou schopni vytvářet události monitorující stisknutí kláves tlačítek a rozumějí oběma výstupům (při stisknutí/při puštění tlačítka).



OBRÁZEK 20 EVENT TLAČÍTKA "W"

Problém, se kterým žáci pracovali byl orientace ve výběru událostí, a to jak v případě výběru z Event dispecheru, tak výběru událostí po stisku klávesy.

Během této části však byly v podstatě všichni žáci schopni vytvořit jak události interagující s tlačítky, tak události interagující s klávesnicí, což se potvrdilo i především v lekcích 5-1 a 6-3.

Z toho důvodu hodnotím splnění tohoto cíle jakožto splněn.

Žák dokáže vytvářet události závislé na interakci s uživatelem																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✓	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✓	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABULKA 11 ŽÁK DOKÁŽE VYTVÁŘET U DÁLOSTI ZÁVISLÉ NA INTERAKCI S U ŽIVATELEM

Žák dokáže spouštět scénáře z událostí, volaných po stisknutí tlačítek na obrazovce či stisknutí tlačítka na klávesnici

Tímto bodem je především myšlena schopnost žáků chápat, jaká událost se spustí, za jaké podmínky (událost „Key W“ po stisku tlačítka „W“, událost „On clicked“ po stisku tlačítka na obrazovce).

Během testování jsem čerpal především z odpovědí na otázky, obsažené v pracovních listech žáků a ze schopnosti žáku zvolit správnou událost při určitém požadavku (Pokud má se má postava pohybovat pomocí tlačítek W, S, A, D, tak zvolím „Keyboard Event [W, S, A, D]“, …). Dále jsem čerpal z otázek v hodině, které jsem pokládal, ve znění „A když bychom potřebovaly, abychom tu postavu ovládali například tlačítky 8, 5, 4, 6, jak to u dělat?“. Ptal jsem se na konci lekce 4-2 a nesetkal jsem se s tím, že by někdo odpověděl špatně.

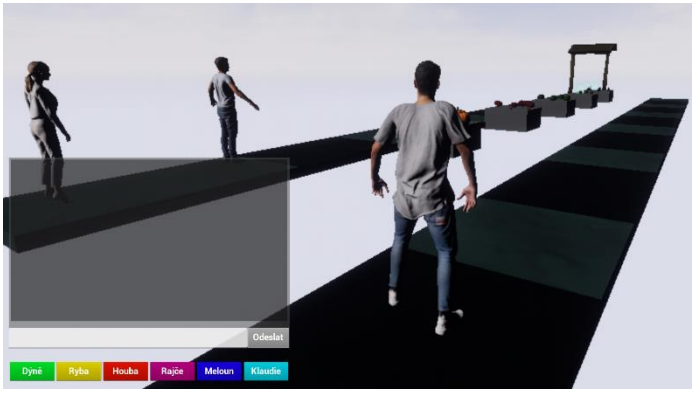
Z informací plynoucí z testování lze konstatovat, že žáci nemají problém s propojením konkrétní události s blokem ve scénáři. Z toho důvodu konstatuji, že daný cíl výuky byl plně splněn.

Žák dokáže spouštět scénáře z událostí, volaných po stisknutí tlačítek na obrazovce či stisknutí tlačítka na klávesnici															
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✓	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✓	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABULKA 12 ŽÁK DOKÁŽE SPOUŠTĚT SCÉNÁŘE Z U DÁLOSTÍ, VOLANÝCH PO STISKNUTÍ TLAČÍTEK NA OBRAZOVCE ČI STISKNUTÍ TLAČÍTKA NA KLÁVESNICI

Tento bod je zaměřen především na schopnost žáků vhodně a stručně pojmenovat dané u dálosti podle toho, jakou činnost vykonávají.

Jako příklad bych rád u vedl úkony z lekce 7-2. V této lekci mají žáci za úkol zajistit, aby postava po stisknutí tlačítka v levém dolním rohu došla k určitému ovoci, sebrala jej a následně se vrátila zpět.



OBRÁZEK 21 U KÁZKA ÚROVNĚ NÁKUP_V2

K tomu, aby byla následná manipulace s postavou možná je

zapotřebí, aby byla tlačítka vhodně a stručně pojmenována podle toho, co vykonávají. V tomto konkrétním případě byla tlačítka pojmenována jednoduše po ovoci, ke kterému se má postava dostat.

Tomuto bodu se věnuje lekce 4-1. Samotné složení lekce se věnuje především naučení postupu, jak vytvořit událost na stisknutí tlačítka a následné manipulace s ním (například právě změna textu) Je tomu tak z důvodu toho, že UE4 k tomuto účelu využívá objekty (každé tlačítko je samo za sebe objekt) a Kvůli tomu jsou operace s tlačítky relativně složité na základní školu.

Jakožto kontrolní mechanismus splnění cíle výuky mi sloužila lekce 7-2, kde bylo vidět, že žáci dokážou jednoduše pojmenovat činnost. Dále pak jsem čerpal z lekce 4-2, kde měly žáci za úkol klávesnicí rozpohybovat kostlivce. Zde jsem se tázal, jak by stručně pojmenovaly danou činnost. Poslední zdroj, odkud jsem ověřoval osvojení si cíle výuky byla celá část 7, kde žáci vytvářeli události a ty musely vhodně pojmenovat.

Závěrem mohu žáci, že žáci až na drobné odchylky byli schopni pojmenovat jednotlivé vstupy. a proto lze považovat cíl výuky za splněný.

Žák dokáže upravit popis vstupů podle činnosti, jež vykoná scénář k nim napojený																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✓	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✓	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABULKA 13 ŽÁK DOKÁŽE U PRAVIT POPIS VSTUPŮ PODLE ČINNOSTI, JEŽ VYKONÁ SCÉNÁŘ K NIM NAPOJENÝ

6.1.5. Část podmínky

Žák dokáže vytvořit podmínku

Daný cíl výuky se zaměřuje na technickou schopnost žáků vložit blok „Když“ (zastupující funkci „If“) do scénáře a připojit k němu blok podmínky (nebo ovládat blok „Když“

zaškrtávacím políčkem). V tomto cíli výuky se tedy neposuzuje pochopení daného bloku, ale pouze jeho zapojení.

Lekce 5-1 se zaměřuje především na představení bloku „Když“, některých základních textových podmínek a způsobu práce s nimi. Později je rozebírána i základní logika jako například „Když odpovíš správně, napiš pochvalu“.

Lekce 5-2 přidává nové prostředí pro podmínky. Jsou zde využívány prostorové podmínky, jako například „Jsem na dosah; Meloun“, která určuje, zda se postava přiblížila k herci „Meloun“, či nikoliv. Zároveň je v této lekci představen koncept opakování s podmínkou.

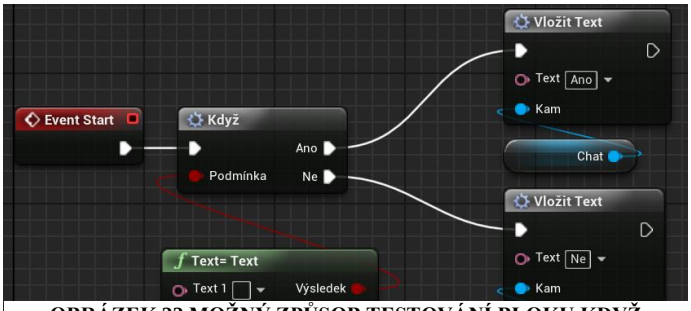
Co se týče schopnosti žáků vytvořit samotnou podmínku, tak byl zaznamenán pouze jediný problém u žáka, který předchozí 2 hodiny chyběl, proto toho musel mnoho dohánět. **Kromě tohoto případu všichni žáci byli technicky schopni podmínku vytvořit, proto považuji cíl výuky za splněný.**

Žák dokáže vytvořit podmínku																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✓	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✓	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABULKA 14 ŽÁK DOKÁŽE VYTVOŘIT PODMÍNKU

Žák dokáže určit kdy je a kdy není podmínka splněna

Jedná se o schopnost žáků vyhodnotit, zda daná jednoduchá logická operace bude vyhodnocena jakožto pravda, či nepravda (1,0; Ano, Ne, …). Příkladem může být vyhodnocení podmínky, kterou vidíte na obrázku vpravo. Z pouhého pohledu na danou podmínku by mělo být žákovy jasné, kdy je podmínka splněna a kdy nikoliv.



OBRÁZEK 22 MOŽNÝ ZPŮSOB TESTOVÁNÍ BLOKU KDYŽ

V tomto konkrétním příkladě je podmínka splněna, když „Text 1 “ a „Text 2“ jsou si rovny neboli, když obsahují stejné znaky ve stejném pořadí.

Pro zkoumání splnění tohoto cíle výuky posloužili především odpovědi na otázky u lekcí 5-1 a 5-2 a dotazování se žáků na každou podmínku, kterou vytvořili při kontrolách dotazem „A ve kterých případech bude signál pokračovat zobáčkem Ano a ve kterých zobáčkem Ne?“.

Základní koncept fungování podmínky pochopili žáci relativně rychle. Když se však přešlo na některé konkrétní podmínky (jako je například ta u vedená výše [problém velkých a malých písmen]) tak žákům nějakou dobu trvalo, než pochopili jejich přesný výsledek. po **několika cílených dotazech na dané podmínky však všichni žáci pochopili jejich funkci, z čehož lze konstatovat, že daný cíl výuky byl splněn.**

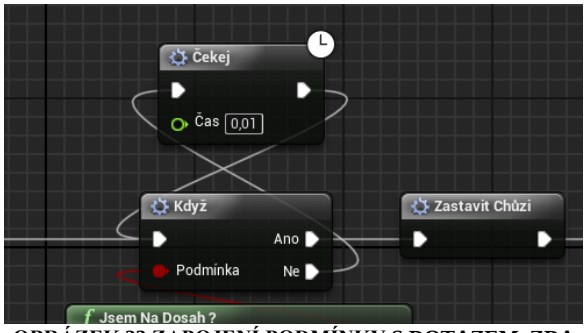
Žák dokáže určit kdy je a kdy není podmínka splněna																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✓	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✓	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABULKA 15 ŽÁK DOKÁŽE U RČIT KDY JE A KDY NENÍ PODMÍNKA SPLNĚNA

Žák dokáže na základě splnění či nesplnění podmínky upravovat scénář za danou podmínkou

V předchozím cíli výuky měli žáci získat schopnost odhalit, kdy je podmínka splněna a kdy ne. Tento cíl výuky přímo navazuje tím, že prověřuje schopnost žáků na základě výsledku podmínky vhodně sestavit scénář za danou podmínkou.

Příkladem může být obrázek vpravo, na kterém lze pozorovat podmínku vyhodnocující, zda se má postava hýbat, či nikoliv.



OBRÁZEK 23 ZAPOJENÍ PODMÍNKY S DOTAZEM, ZDA JE POSTAVA NA DOSAH S POUŽITÍM OPAKOVÁNÍ

Při vyhodnocování tohoto cíle výuky jsem bral v potaz stejné kritéria jako z předchozího cíle výuky a k tomu jsem přidal způsob kontroly, kdy žáci měli za úkol jet prstem po bílé čáře v jejich scénáři podle splnění, či nesplnění podmínky, kterou jsem jim zadal („Ukaž, když postava nestojí u Melounu, jak pojede signál po bílé čáře“).

Jedna skupina měla ze začátku výrazný problém s identifikací dráhy pohybu signálu, avšak v následujících hodinách po opakovaném tréningu „jízdy prstem po čáře“ byla i tato skupina schopna určit, jak musí scénář pokračovat. Z tohoto důvodu vyhodnocuji daný cíl výuky za splněný.

Žák dokáže na základě splnění či nesplnění podmínky upravovat scénář za danou podmínkou																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
[1] Erik I	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐	☑	☐
[2] Erik II	☑	☑	☐	☑	☑	☑	☑	☑	☐	☐	☑	☑	☑	☑	☐	☑
[2] Ondřej	☑	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[3] Vojtěch	☑	☑	☑	☐	☑	☑	☐	☑	☑	☑	☐	☑	☑	☑	☐	☑
[3] Aneta	☑	☐	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[4] Eliška	☑	☑	☐	☑	☐	☐	☑	☑	☑	☑	☑	☐	☑	☐	☐	☐
[4] Lucie	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐	☑	☑	☑
[4] Nikola	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

TABULKA 16 ŽÁK DOKÁŽE NA ZÁKLADĚ SPLNĚNÍ ČI NESPLNĚNÍ PODMÍNKY U PRAVOVAT SCÉNÁŘ ZA DANOU PODMÍNKOU

6.1.6. Část parametry

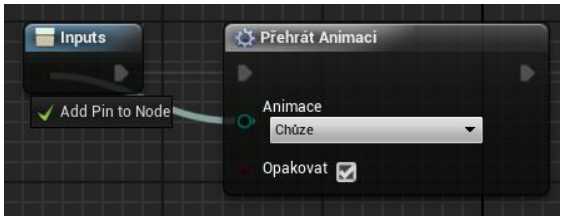
Žák dokáže přidat do vlastních bloků parametry a následně je využívat

Jedná se o schopnost žáků vytvořit jednotlivé parametry ve vlastních blocích.

Samotná dovednost se skládá ze dvou částí:

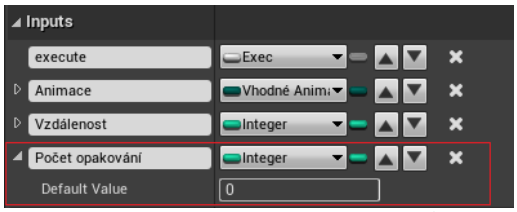
První část spočívá v tom vytvořit samotné parametry. UE4 nabízí 2 možnosti.

- První možnost je samotné vytvoření parametru při tvorbě bloku. Tento způsob je efektivní při samotném tvoření bloku na počátku (neboli bez scénáře v něm obsaženém), avšak obsahuje poměrně hodně kroků, které ještě žáci v této fázi nedokážou obsáhnout.
- Druhá možnost spočívá v prostém přetáhnutí potřebného parametru do vstupu bloku viz obrázek vpravo. Jedná se o efektivní způsob přidávání parametrů za běhu fyzického vývoje (při samotné tvorbě scénáře v bloku) a zároveň je velmi jednoduchý a intuitivní, tudíž se žáci učí tento způsob.



OBRÁZEK 24 VYTVÁŘENÍ PARAMETRU

Druhá část spočívá ve vhodném pojmenování jednotlivých parametrů. Zde žáci pracují v tabulce, viz obrázek v pravo.



OBRÁZEK 25 DETAIL PARAMETRŮ

Lekce 6-1 se věnuje především představení konceptu parametrů. V této lekci je žákům představena výhoda parametrů v podobě možnosti modifikace bloku bez nutnosti vytvářet blok nový.

Lekce 6-2 slouží především k procvičení vytváření parametrů v pro žáky již známém prostředí. u této lekce je chybně specifikována formulace úkolu pro žáky. Z toho důvodu se stávalo, že žákům bylo potřeba dovysvětlit, jak danný úkol byl myšlen. po vysvětlení byly však žáci schopni úkol splnit.

Lekce 6-3 měla sloužit pčedevším pro komplexnější zapracování parametrů do vlastnícch bloků v úloze věnující se chatu. Tato lekce se však u kázala jakožto nejsložitější z celé sady úloh a správně ji dokázaly dokončit jen 2 skupiny (zbylé skupiny ji chtěli dokončit též, a tak ji dokončovali kolektivně). Z toho důvodu tato lekce vyžaduje úpravy.

Schrnutím lze říci, že největší problém, který žáci řešily bylo vhodné pojmenování parametrů. Nebylo neobvyklé, že žáci pojmenovali parametr např. A,B,C nebo 1,2,3 a bylo potřeba se s žáky zamyslet nad tím, zda je takovéto pojmenování vhodné.

Shrnutím je možné konstatovat že ačkoliv nemám výslekdy od jedné žákyně (z důvodu její následné absence až do konce testování), tak zbylí žáci byli schopni vytvářet a pracovat s parametry, tudíš považuji daný cíl výuky za splněný.

Žák dokáže přidat do vlastních bloků parametry a následně je využívat																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✓	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✓	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

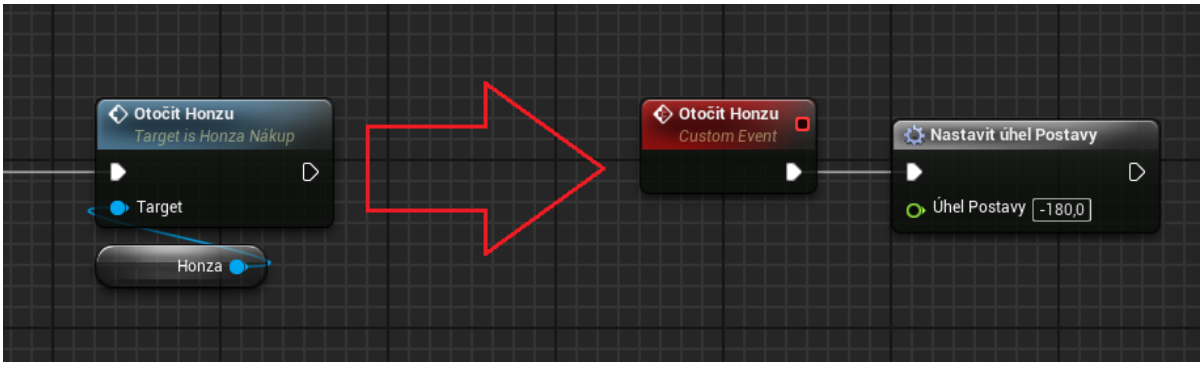
TABULKA 17 ŽÁK DOKÁŽE PŘIDAT DO VLASTNÍCH BLOKŮ PARAMETRY A NÁSLEDNĚ JE VYUŽÍVAT

6.1.7. Část volané události

Žák dokáže vytvářet a používat volané události pro komunikaci mezi objekty

Jedná se o dovednost žáků komunikovat mezi různými objekty u místěnými ve světě.

K tomuto účelu má UE4 obsažen několik nástrojů. Nejjednodušší z nich je takzvaný „Custom_Event“, neboli volaná událost. Tento prvek funguje obdobně jakožto „Macro“ využívané pro vlastní bloky žáků s tím rozdílem, že je volán přímo do hlavního plánu (viz obrázek níže) a především je možné jej volat do jiného objektu.



OBRÁZEK 26 PRINCIP VOLANÝCH U DÁLOSTÍ

Lekce 7-1 je zaměřena především na tvorbu těchto volaných událostí. Zaměřuje se na samotné vyhledání volané události, její pojmenování, připojení proměnné tipu object a vysvětlení její funkce a samotné zprovoznění volané události.

Lekce 7-2 se snaží klást více důrazu na samostatné přemýšlení žáků. Cílem lekce je zautomatizování procesu vytváření volaných událostí a skládání složitějších struktur.

Po testování mohu suverénně tvrdit, že volané události jsou pro žáky nejsložitější prvek na vytvoření, se kterým žáci pracovaly. Jejich tvorba je poměrně složitá a nepodporuje následné chápání jejich funkcionality.

Někteří žáci pochopily princip, jak volané události fungují, ale nedokázaly je bez asistence vytvořit. Jiní žáci naopak přesně věděli, jak dané události vytvořit, ale nevěděli, jak s nimi pracovat. Z toho důvodu nemůže být daný cíl výuky označen za splněný.

Žák dokáže vytvářet a používat volané události pro komunikaci mezi objekty																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[1] Erik I	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	☐
[2] Erik II	✓	✓	☐	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓	✓	☐	✓
[2] Ondřej	✓	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[3] Vojtěch	✓	✓	✓	☐	✓	✓	☐	✓	✓	✓	☐	✓	✓	✓	☐	✓
[3] Aneta	✓	☐	✓	✓	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	☐	✓
[4] Eliška	✓	✓	☐	✓	☐	☐	✓	✓	✓	✓	✓	☐	✓	☐	☐	☐
[4] Lucie	✓	☐	✓	✓	✓	✓	✓	✓	✓	✓	✓	☐	☐	✓	✓	✓
[4] Nikola	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABULKA 18 ŽÁK DOKÁŽE VYTVÁŘET A POUŽÍVAT VOLANÉ U DÁLOSTI PRO KOMUNIKACI MEZI OBJEKTY

6.1.8. Část proměnné

Žák dokáže vytvářet a používat proměnné

Jedná se především o schopnosti vytvořit proměnou, alespoň typově chápat základní datové typy a znát použití proměnné v módu get a set (neboli u mět do proměnné u kládat a číst hodnoty).

Lekce 8-1 se zaměřuje na u kázání technické realizace výše zmíněných schopností. na počátku jsou shrnuty datové typy a následně žáci vytvářejí proměnné.

Lekce 8-2 navazuje a snaží se propojit povědomí o proměnných s jinými prvky, v tomto případě konkrétně s volanými událostmi.

První lekce proběhla bez problémů a žáci byli technicky schopni vytvořit proměnné a využívat je.

Druhá lekce však žákům vůbec nešla, především pak kvůli nepochopení volaných u dálostí.

Problematické bylo také to, že v době, kdy se správně měla řešit lekce 8-1 byla většina žáků nepřítomna, Kvůli čemuž nebylo možné plně otestovat porozumění proměnným. Z těchto důvodu u dávám daný cíl výuky jakožto neplně ověřený.

Žák dokáže vytvářet a používat proměnné																
Hodina č.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
[1] Matouš	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
[1] Erik I	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐	☑	☐
[2] Erik II	☑	☑	☐	☑	☑	☑	☑	☑	☐	☐	☑	☑	☑	☑	☐	☑
[2] Ondřej	☑	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[3] Vojtěch	☑	☑	☑	☐	☑	☑	☐	☑	☑	☑	☐	☑	☑	☑	☐	☑
[3] Aneta	☑	☐	☑	☑	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☐	☑
[4] Eliška	☑	☑	☐	☑	☐	☐	☑	☑	☑	☑	☑	☐	☑	☐	☐	☐
[4] Lucie	☑	☐	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐	☑	☑	☑
[4] Nikola	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑

TABULKA 19 ŽÁK DOKÁŽE VYTVÁŘET A POUŽÍVAT PROMĚNNÉ

6.1.9. Shrnutí výsledků ověřování cíle výuky

- Žák dokáže spustit projekt vytvořený v UE4

  Většina žáků byla schopná hned po první hodině projekt opětovně spustit. Zbylí žáci toho byli schopni hned v následujících hodinách. Celkově tedy lze konstatovat že tento cíl výuky byl u všech žáků naplněn.

- Žák dokáže sestavit scénář, jímž splní zadaný úkol>> Žák rozumí důvodu pořadí bloků, a dokáže plnit jednoduché úkoly

  Většina žáků byla po lekcích určených k naučení základní práce se scénářem schopna bezproblémově pracovat, tudíž tento cíl výuky lze považovat za splněný.

- Žák dokáže scénář upravovat mazáním bloků, či změnou jejich pořadí

  Všichni žáci velice rychle pochopili, jak je efektivně mazat a měnit jejich pořadí čili tento cíl výuky považuji za splněný.

- Žák dokáže spustit simulaci a ověřit správnost svého scénáře

  Kromě drobného problému s tlačítkem „Simulation“, který se vyřešil v první části lze konstatovat, že jsem nezaznamenal žádný případ, kdyby žáci nevěděli,

jak spustit simulaci a tím si ověřit funkčnost svého kódu. s ohledem na to konstatuji že tento cíl výuky byl naplněn.

- Žák ví, jak funguje opakování

  Ačkoliv lekce 2-2 nevyšla dle očekávání, tak díly lekci 2-3 mohu konstatovat, že žáci byli schopni pochopit, jak opakování v programování funguje. Díky tomu mohu u soudit, že tento daný cíl výuky je splněn.

- Žák pozná, kdy je vhodné použít opakování

  Z těchto dat nelze říct, že si žáci plně osvojili schopnost rozpoznat, zda opakování intuitivně využít, či nikoliv. na druhou stranu žáci jsou schopni po absolvování těchto lekcí opakování v triviálních případech využívat. Z toho důvodu jsou data k této otázce vyhodnocena ve většině případů jakožto neprůkazné.

- Žák dokáže do scénáře, vložit opakování, pokud jej scénář neobsahuje

  Při testování tohoto cíle výuky žáci z počátku občasně zapomínali na podstatné dovednosti, avšak po několika vytvořených opakováních se žáci všem podstatným dovednostem naučili. Kvůli tomu mohu konstatovat, že daný cíl výuky byl naplněn.

- Žák dokáže vytvořit a pojmenovat vlastní blok

  Po úspěšném zvládnutí první lekce této části již žáci neměli problém s vytvářením vlastních bloků, ač jsem předpokládal, že postup jejich tvorby část žáků zapomene. Z tohoto důvodu hodnotím tento cíl výuky jakožto splněný.

- Žák dokáže se svými bloky pracovat ve scénáři

  Žáci jsou schopni s vlastními bloky pracovat podle instrukcí, avšak ne intuitivně. Z toho důvodu lze konstatovat, že cíl výuky byl splněn částečně.

- Žák dokáže vytvářet události závislé na interakci s uživatelem

  Během této části byly v podstatě všichni žáci schopni vytvořit jak události interagující s tlačítky, tak události interagující s klávesnicí. Z toho důvodu hodnotím splnění tohoto cíle jakožto splněn.

- Žák dokáže spouštět scénáře z událostí, volaných po stisknutí tlačítek na obrazovce či stisknutí tlačítka na klávesnici

  Žáci nemají problém s propojením konkrétní události s blokem ve scénáři. Z toho důvodu konstatuji, že daný cíl výuky byl plně splněn.

- Žák dokáže u pravit popis vstupů podle činnosti, jež vykoná scénář k nim napojený

  Žáci až na drobné odchylky byli schopni pojmenovat jednotlivé vstupy. a proto lze považovat cíl výuky za splněný.

- Žák dokáže vytvořit podmínku

  Všichni žáci byli technicky schopni podmínku vytvořit, proto považuji cíl výuky za splněný.

- Žák dokáže určit kdy je a kdy není podmínka splněna

  Po několika cílených dotazech žáci pochopili funkci podmínek, z čehož lze konstatovat, že daný cíl výuky byl splněn.

- Žák dokáže na základě splnění či nesplnění podmínky upravovat scénář za danou podmínkou

  Jedna skupina měla ze začátku výrazný problém s identifikací dráhy pohybu signálu, avšak v následujících hodinách po opakovaném tréninku „jízdy prstem po čáře“ byla i tato skupina schopna určit, jak musí scénář pokračovat. Z tohoto důvodu vyhodnocuji daný cíl výuky za splněný.

- Žák dokáže přidat do vlastních bloků parametry a následně je využívat

  Žáci byli schopni vytvářet a pracovat s parametry, tudíš považuji daný cíl výuky za splněný.

- Žák dokáže vytvářet a používat volané události pro komunikaci mezi objekty

  Někteří žáci pochopili princip, jak volané události fungují, ale nedokázali je bez asistence vytvořit. Jiní žáci naopak přesně věděli, jak dané události vytvořit, ale nevěděli, jak s nimi pracovat. Z toho důvodu nemůže být daný cíl výuky označen za splněný.

- Žák dokáže vytvářet a používat proměnné

  Nebylo možné plně otestovat porozumění proměnným. Z toho důvodu u dávám daný cíl výuky jakožto neplně ověřený.

Cíl výuky	Výsledek ověřování
· Žák dokáže spustit projekt vytvořený v UE4	**naplněno**
· Žák dokáže sestavit scénář, jímž splní zadaný úkol	**naplněno**
· Žák dokáže scénář upravovat mazáním bloků, či změnou jejich pořadí	**naplněno**
· Žák dokáže spustit simulaci a ověřit správnost svého scénáře	**naplněno**
· Žák ví, jak funguje opakování	**naplněno**
· Žák pozná, kdy je vhodné použít opakování	**neprůkazné**
· Žák dokáže do scénáře, vložit opakování, pokud jej scénář neobsahuje	**naplněno**
· Žák dokáže vytvořit a pojmenovat vlastní blok	**naplněno**
· Žák dokáže se svými bloky pracovat ve scénáři	**naplněno**
· Žák dokáže vytvářet události závislé na interakci s uživatelem	**naplněno částečně**
· Žák dokáže spouštět scénáře z událostí,...	**naplněno**
· Žák dokáže upravit popis vstupů podle činnosti, jež vykoná scénář k nim napojený	**naplněno**
· Žák dokáže vytvořit podmínku	**naplněno**
· Žák dokáže určit kdy je a kdy není podmínka splněna	**naplněno**
· Žák dokáže na základě splnění či nesplnění podmínky upravovat scénář...	**naplněno**
· Žák dokáže přidat do vlastních bloků parametry a následně je využívat	**naplněno**
· Žák dokáže vytvářet a používat volané události pro komunikaci mezi objekty	**nesplněný**
· Žák dokáže vytvářet a používat proměnné	**neplně ověčený**

TABULKA 20 TABULKA VÝSLEDKŮ OVĚŘOVÁNÍ CÍLŮ VÝUKY

6.2. Srovnávání zlepšení informatického myšlení

Ke zjištění toho, zda u žáků bylo rozvinuto informatické myšlení, bylo využito srovnávacích testů ze sady otázek Bobříka informatiky.

Jedná se o předmětovou soutěž podporovanou ministerstvem školství a klade si za cíl seznamovat žáky a jejich učitele s informatickými otázkami a problémy, tedy přiblížit moderní předmět „Informatika“. Soutěžící odpovídají na otázky z oblasti informatického myšlení, algoritmizace, porozumění informacím, řešení problémů a digitální gramotnosti. Soutěž probíhá v několika kategoriích, které jsou rozděleny podle věku. [19]

Volba této sady otázek byla provedena po jejím prostudování a konzultaci s vedoucím práce. Dalším důvodem bylo prostudování dokumentu, který se věnuje důvodu vzniku Bobříka informatiky a charakterem celé soutěže, jenž je vhodný pro ověřování úrovně osvojení informatického myšlení. [20]

Ze soutěže Bobřík informatiky bylo vybráno deset otázek, které se týkají informatického myšlení. Otázky pochází z kategorií odpovídající druhému stupni ZŠ. Tyto otázky byly následně rozděleny do dvou testů po pěti (oba testy jsou součástí přílohy). Žáci vyplňovali oba

testy, jeden při zahájení ověřování úloh, druhý po završení ověřování. Krom žáků, kteří byli součástí ověřování sady úloh testy vyplňovali žáci sedmé třídy základní školy Choustník. Výsledky od těchto žáků sloužily jakožto data od kontrolní skupiny.

Ideální stav, kterého jsem se pokoušel dosáhnout jak při zjištění úrovně zlepšení informatického myšlení, tak při samotném ověřování úloh byl takový, kdy bych provedl srovnávací výsledky na třech školách paralelně, a to sice v Choustníku, v Soběslavi a v Táboře. Tato varianta by vedla ke zvýšení relevantnosti porovnávání. Se všemi školami jsem byl již domluven na spolupráci, avšak opětovně z důvodu onemocnění Covid-19 většina škol nebyla ochotna dovolit zahájení kroužku programování. Z toho důvodu mohlo proběhnout porovnávání pouze na škole v Choustníku, kde mi vše bylo umožněno.

6.2.1. Výsledky srovnání

V experimentální skupině bylo započteno všech devět žáků, kteří byli započteni v ověřování cílů výuky. V kontrolní skupině bylo započteno 15 žáků, kteří se účastnili obou testů.

Úvodní test - experimentální skupina						
Žáci	Otázky					Správných odpovědí
	1	2	3	4	5	
[1] Matouš	✓	✓	✓	✓	✓	5
[1] Erik I	✓	✓	✓		✓	4
[2] Erik II	✓		✓			2
[2] Ondřej	✓	✓		✓	✓	4
[3] Vojtěch	✓		✓			2
[3] Aneta	✓	✓		✓		3
[4] Eliška	✓	✓	✓			3
[4] Lucie			✓			1
[4] Nikola	✓	✓	✓	✓	✓	5
Celkem:	8	6	7	4	4	29
Průměr na žáka:						3,2

TABULKA 21 VÝSLEDKY EXPERIMENTÁLNÍ SKUPINY ÚVODNÍ TEST

Úvodní test - kontrolní skupina						
Žáci	Otázky					Správných odpovědí
	1	2	3	4	5	
Žák č.1	✓	✓		✓		3
Žák č.2	✓	✓	✓	✓	✓	5
Žák č.3		✓	✓			2
Žák č.4	✓			✓	✓	3
Žák č.5	✓	✓				2
Žák č.6	✓	✓	✓			3
Žák č.7			✓	✓		2
Žák č.8	✓	✓				2
Žák č.9	✓	✓	✓	✓	✓	5
Žák č.10	✓		✓	✓		3
Žák č.11				✓		1
Žák č.12	✓	✓				2
Žák č.13		✓		✓		2
Žák č.14	✓		✓		✓	3
Žák č.15	✓	✓	✓			3
Celkem:	11	10	8	8	4	41
Průměr na žáka:						2,7

TABULKA 22 VÝSLEDKY KONTROLNÍ SKUPINY ÚVODNÍ TEST

Ze zjištěných dat vyplívá, že experimentální skupina disponuje lepším průměrem na žáka než skupina. Důvodem může být to, že žáci

v experimentální skupině jsou věkově smíšeni. Dalším důvodem může být skutečnost, že testování probíhalo v rámci zájmového kroužku, což mohlo vést k selekci žáků s většími predispozicemi k informatickému myšlení. Ze zjištěných dat lze vyčíslit koeficient poměru průměru na žáka experimentální a kontrolní skupiny na 1,17.

Závěrešný test - experimentální skupina						
Žáci	Otázky					Správných odpovědí
	1	2	3	4	5	
[1] Matouš	☑	☑	☑	☐	☑	4
[1] Erik I	☑	☑	☑	☑	☑	5
[2] Erik II	☑	☐	☑	☑	☐	3
[2] Ondřej	☑	☑	☐	☐	☑	3
[3] Vojtěch	☑	☑	☑	☐	☐	3
[3] Aneta	☑	☐	☑	☑	☑	4
[4] Eliška	☑	☑	☑	☑	☐	4
[4] Lucie	☑	☐	☐	☑	☑	3
[4] Nikola	☑	☐	☑	☑	☑	4
Celkem:	9	5	7	6	6	33
Průměr na žáka:						3,7

TABULKA 23 VÝSLEDKY EXPERIMENTÁLNÍ SKUPINY ZÁVĚREČNÝ TEST

Závěrečný test - kontrolní skupina						
Žáci	Otázky					Správných odpovědí
	1	2	3	4	5	
Žák č.1	☑	☑	☑	☐	☑	4
Žák č.2	☑	☐	☑	☑	☐	3
Žák č.3	☑	☑	☐	☐	☑	3
Žák č.4	☑	☐	☐	☑	☑	3
Žák č.5	☐	☐	☐	☐	☐	0
Žák č.6	☑	☑	☐	☑	☑	4
Žák č.7	☐	☐	☑	☑	☐	2
Žák č.8	☑	☐	☐	☐	☐	1
Žák č.9	☑	☑	☑	☐	☐	3
Žák č.10	☑	☑	☐	☑	☐	3
Žák č.11	☐	☐	☐	☑	☑	2
Žák č.12	☑	☑	☑	☐	☐	3
Žák č.13	☐	☐	☐	☑	☐	1
Žák č.14	☑	☐	☐	☐	☑	2
Žák č.15	☑	☐	☑	☐	☑	3
Celkem:	11	6	6	7	7	37
Průměr na žáka:						2,5

TABULKA 24 VÝSLEDKY KONTROLNÍ SKUPINY ZÁVĚREŠNÝ TEST

Ze zjištěných dat lze pozorovat několik skutečností. První z nich je podobný poměr průměru na žáka u kontrolní skupiny, pokud bychom porovnávali úvodní test a závěrečný test, tak se jedná o mírný pokles z 2,7 na 2,5. Tento pokles není nijak rapidní, takže se může jednat o statistickou odchylku.

Druhá skutečnost je taková, že průměr na žáka experimentální skupiny vzrostl z 3,2 na 3,7. Zde se jedná již o nezanedbatelný nárůst.

Z těchto dat lze vypočítat nový koeficient poměru průměru na žáka experimentální a kontrolní skupiny na 1,49. Vzhledem k původnímu koeficientu (1,17) se jedná o zlepšení o 0,31.

6.2.2. Shrnutí výsledků ověřování zlepšení úrovně informatického myšlení

Z dat, která se podařilo hmatatelně nashromáždit lze pozorovat, že ke zlepšení informatického myšlení po absolvování ověřované sady úloh skutečně došlo.

Samotné pozorování mohlo ovlivnit několik faktorů:

- Dlouhá prodleva mezi jednotlivými hodinami.
- Obecná specifická situace (žáci přistupovali ke škole jiným způsobem).
- Výsledky pouze z jedné školy.

Naproti těmto faktorům byla u žáků pozorována zajímavá skutečnost. Většina žáků jeví po absolvování tohoto testování zvýšený zájem o informatiku a programování. Tři žáci, kteří nebyli zcela rozhodnuti, jaký obor chtějí na střední škole studovat deklarovali, že chtějí studovat obory spojené s informatiko (2x programátor, 1x obecná informatika). Všech devět žáků si přeje pokračovat v daném kroužku a ponořit se hlouběji do samotného systému Unreal Engine 4 (popřípadě Unreal Engine 5). Dále zhruba polovina žáků má systém nainstalován doma a pokouší se vytvořit si vlastní hru. Poslední zajímavé zjištění je zvýšený zájem o pokračování v kroužku jak od spolužáků dětí v experimentální skupině, tak od vedení školy, neboť dle slov vedení školy kroužek přispívá ke zlepšení dobrého jména ZŠ Choustník.

Z těchto důvodů lze konstatovat, že daná sada úloh přispívá ke zlepšení úrovně osvojení informatického myšlení.

7. Závěr

Popsal jsem prostředí Unreal Engine 4 a systém vizuálního skriptování Blueprint, který je využíván jakožto alternativa ke klasickému programovacímu jazyku. Při popisu jsem využil texty, které jsem již na toto téma zpracovával.

Představil jsem pojem informatické myšlení, spolu s jeho složkami. Při tom jsem popsal jeho princip, důležitost a užitečnost v každodenním životě. u každé ze složek jsem provedl popis a zároveň jsem uvedl příklady využití každé složky v reálném životě a při vývoji videohry. na závěr jsem vybral ty ze složek, které se programováním rozvíjejí nejvíce.

Provedl jsem analýzu úloh z pohledu toho, zda se v úlohách vyskytují podněty pro rozvíjení vybraných složek informatického myšlení. Z proběhlé analýzy bylo zjištěno, že úlohy skutečně tyto podněty obsahují, ač v různých měřítkách.

Teoreticky jsem popsal metodu akční výzkum. Z prozkoumané literatury bylo zjištěno, že existují různé varianty realizace, tudíž termín akční výzkum spíše zaštiťuje vícero metod zkoumání.

Provedl jsem ověření úloh z pohledu naplnění cílů výuky. Ze získaných dat lze potvrdit naplnění většiny cílů výuky. Cíle výuky, které nebyly potvrzeny celkem čtyři. u těchto výsledků byly vysvětleny důvody jejich nenaplnění.

Dále bylo provedeno srovnání zlepšení osvojení informatického myšlení a změna zájmu o obor informatika u žáků, kteří absolvovali testování sady úloh. Ze zjištěných výsledků vyplynulo, že ke zlepšení informatického myšlení skutečně došlo a u žáků lze pozorovat zvýšený zájem o obor informatika.

Jako nejzajímavější zjištění práce považuji skutečnost, že si žáci po absolvování dané sady úloh z pravidla oblíbí obor informatika. Tato skutečnost může být zapříčiněna i metodou akčního výzkumu, neboť i já sám jsem o informatice pojednával s nadšením.

Samotná práce může být v budoucnu využita jakožto příspěvek k debatě o začlenění prvků videoher do obecné výuky. Ze zjištěných informací je patrné, že si žáci oblíbili software Unreal Engine 4 také z důvodu možného budoucího využití v profesionálních skupinách.

Na tuto práci lze navázat několika způsoby. Jedna z možností je na základě zjištěných dat vylepšit sadu úloh tak, aby byly skutečně všechny cíle výuky naplněny. Dále jo možné provést výzkum dané sady úloh jinými výzkumnými metodami, pro potvrzení či vyvrácení zjištěných informací.

Poděkování: výzkum v této práci byl částečně hrazen z prostředků projektu GAJU 041/2022/S "Klíčová místa kurikula pro integraci vzdělávacích obsahů v oblasti STEM".

Seznam použité literatury a zdrojů

[1] PACLT, D., *Tvorba projektů pro výuku programování v prostředí u nreal Engine 4* [online]. Dostupný z WWW: https://wstag.jcu.cz/portal/studium/prohlizeni.html?pc_pagenavigationalstate=AAAAAQAGMjI5NTkxEwEAAAABAAhzdGF0ZUtleQAAAAEAFC05MjIzMzcyMDM2ODU0NzcyOTkxAAAAAA**#prohlizeniSearchResult

[2] DENNING, P. J., M. TEDRE. Computational Thinking. Cambridge: The MIT Press, 2019. ISBN 978026253656

[3] *Co je informatické myšlení* [online]. Jihočeská u niverzita v Českých Budějovicích. Dostupný z WWW: https://imysleni.cz/informatickemysleni/co-je-informaticke-mysleni

[4] MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE a TĚLOVÝCHOVY. *Strategie digitálního vzdělávání* [online]. Dostupný z WWW: https://www.msmt.cz/vzdelavani/skolstvi-v-cr/strategie-digitalniho-vzdelavani-do-roku-2020

[5] THE CHARACTED INSTITUTE FOR IT. *Výpočetní myšlení CAS – Průvodce pro u čitele*. [online]. Dostupný z WWW: https://community.computingatschool.org.uk/resources/2324/single

[6] VANÍČEK, J., et al. *Programování ve Scratch pro 2. stupeň základní školy* [online]. Jihočeská u niverzita v Českých Budějovicích. Dostupné z WWW: https://www.imysleni.cz/ucebnice/programovani-ve-scratchi-pro-2-stupen-zakladni-skoly

[7] LIUKAS, Linda. Hello Ruby. *Dobrodružné programování* [online]. Přeložil Ondřej NOVÁK. Praha: Dynastie, 2017. ISBN 9788090580343. Dostupný z WWW: https://www.databazeknih.cz/knihy/hello-ruby-dobrodruzne-programovani-372488

[8] CHALOUPKOVÁ, Věra. *Předpoklady a možnosti rozvoje matematických a informatických představ v raném věku s využitím robotických hraček* [online]. České Budějovice, 2018. Dostupné z WWW: https://wstag.jcu.cz/portal/studium/prohlizeni.html?pc_pagenavigationalstate=AAAAAQAGMjI2ODUzEwEA

[9] BERRY, M. *Computational Thinking in Primary Schools* [online]. Dostupné z WWW: http://milesberry.net/2014/03/computational-thinking-in-primary-schools/

[10] LEE, J., *Learning u nreal Engine Game Development* [online]. Packt Publishing Limited, 2016. Dostupné z WWW: https://subscription.packtpub.com/book/game_development/9781784398156/1/ch01lvl1sec10/the-history-of-unreal-engine>.ISBN: 9781784398156

[11] DICKINS, Rosie. *Počítače a programování: podívej se pod okénko*. Ilustroval Shaw NIELSEN, přeložil Jaroslav KUČERA. Praha: Svojtka & Co., 2016. Podívej se pod okénko. ISBN 978-80-256-1793-9.D

[12] Projekt PRIM — informace k ověřování 2019/2020 [online]. Dostupné z WWW: https://popelka.ms.mff.cuni.cz/~lessner/mw/index.php/U%C4%8Debnice/PRIM?tgb-teacher=true

[13] EDUSKOP. *PRIM-02 Jak rozvíjet informatické myšlení* [online]. Dostupné z WWW: https://eduskop.cz/courses/course-v1:UWB+PRIM-02+2020/courseware/4a63d5a35e634d94b83c62edbbe72f8b/9d8767b0a7e1414a83c202136c1c084f/?child=first

[14] PECINOVSKÝ, Rudolf. *Současné trendy v metodice výuky programování. Česká škola* [online]. Dostupný z WWW: http://www.ceskaskola.cz/ICTveskole/Ar.asp?ARI=102900&CAI=2129

[15] FUTSCHEK. *Algorithmic Thinking: The Key for u nderstanding Computer Science*

[16] LANKSHEAR,C., KNOBEL, M. a *handbook of teacher research. McGraw-Hill International*. 2007. ISBN 978-03-352-2610-8.

[17] LEWIN, K. Action research and minority problems. Journal od Social Issues. 1946. ISSN 1540-4560.

[18] HABERMAS, J. *Knowledge and human interests. London: Heinemann,* 1972, 368 s., ISBN 0-807-01541-5.

[19] Bobřík informatiky [online]. Dostupné z WWW: https://www.ibobr.cz/o-soutezi/obecne-informace

[20] VANÍČEK, J. *Bobřík informatiky, soutěž žáků a studentů v informatice* [online]. Dostupné z WWW: https://www.bebras.org/sites/default/files/documents/publications/Vanicek-2008.pdf

Přílohy

1. Žáky absolvované sady úloh – projekty a pracovní listy
2. Testy pro porovnání osvojení informatického myšlení.

Tyto přílohy jsou z důvodu jejich velikosti součástí pouze elektronické verze práce na paměťové kartě, kde je též možné nalézt diplomovou práci ve formátu PDF.