https://doi.org/10.53656/math2026-6-4-tap

Educational Technologies
Образователни технологии

ТЕОРЕТИЧЕН АНАЛИЗ НА ВЪЗМОЖНОСТИТЕ ЗА ИЗПОЛЗВАНЕ НА ВИЗУАЛНО ПРОГРАМИРАНЕ С KODU В ЧАСОВЕТЕ ПО КОМПЮТЪРНО МОДЕЛИРАНЕ В 3. И 4. КЛАС

Цветанка Георгиева-Трифонова[1)],
Петя Григорова-Калчева[2)],
[1)]Великотърновски университет „Св. св. Кирил и Методий“ (България)
[2)]Софийски университет „Св. Климент Охридски“ (България)

Резюме. В настоящата работа са проучени възможностите за употреба на средата за визуално програмиране Kodu като алтернатива на утвърдените среди за блоково програмиране, предлагащи пропедевтика за най-малките ученици. В контекста на учебните програми в България е потвърдена възможността за използване на визуално програмиране с Kodu като ефективен инструмент за обучение по компютърно моделиране в 3. и 4. клас. Направен е анализ на езика за правила Semantic Web Rule Language (SWRL) и е представен илюстративен пример за онтология в Kodu. Проучването показва, че правилата за визуално програмиране в Kodu могат да се сравнят със SWRL правила. Това предоставя ценни аналитични възможности за разширяване на учебните практики към областта на изкуствения интелект. По този начин средата Kodu дава известно предимство, защото може да се използва не само за първи стъпки в програмирането, но може послужи и като пропедевтика за бъдещо обучение за работа с изкуствен интелект.

Ключови думи: визуално програмиране; Kodu; изкуствен интелект

1. Въведение

Средите за визуално програмиране (и по-конкретно блоково програмиране като Scratch) играят важна роля като първа стъпка в програмирането. Благодарение на достъпния си интерфейс и игровия

подход те позволяват на учениците да се запознаят с концепции като условни изрази, цикли, променливи и подпрограми. *Чрез компютърно моделиране се очаква децата да бъдат мотивирани, че програмирането е приятна професия – изключително лесно се създават много ефектни игри и анимации, а от друга страна, започва формирането на т.нар. алгоритмично мислене у тях* (Petrov, 2021). Традиционно те се използват като пропедевтика към изучаването на по-сложни процедурни или обектно ориентирани езици като C, C++, C#, Java и Python, които се използват широко в индустрията и академичните среди (Chotova, 2021). Постепенният преход от визуално към текстово програмиране улеснява адаптацията на учениците към по-високи нива на сложност в програмирането.

Подобно на Scratch, Kodu също е платформа за визуално програмиране, която е специално проектирана за деца. Тя предоставя интуитивен интерфейс, базиран на визуални блокове, което улеснява разбирането на основни програмни концепции без необходимост от писане на код. Основава се на правила и поведения, което я прави подходяща за обучение по логическо мислене и структурирано решаване на проблеми (Coy, 2013; Kelly, 2013; Marghitu & Coy, 2015). Kodu използва паралелна система от правила, която стимулира обучаемите да мислят алгоритмично, докато създават собствени проекти[1] (MacLaurin, 2009).

В България навлизането на Kodu Game Lab в училищната практика се свързва с дейността на Ангел Ангелов, който чрез инициативата „KODU България“[2] популяризира възможностите на Kodu за използване в началния етап на обучение. Неговата работа е насочена към предоставяне на обучителни ресурси, методически указания и практически примери за използване на Kodu като средство за компютърно моделиране и програмиране в начален етап на обучение; разглеждане на мястото и ролята на компютърните игри в класната стая и спецификите на визуалното програмиране с Kodu като средство за развитие на логическо мислене, креативност и дигитална грамотност при учениците (Angelov & Momcheva, 2012), (Angelov et al., 2012a), (Angelov et al., 2012b).

Законите на изчисленията, които се изучават в подобни среди, играят ключова роля за разбирането на програмите. Те позволяват на обучаемите да предсказват поведението на дадена програма, да откриват грешки и да изграждат по-сложни разработки, които функционират правилно. Езикът Calypso, разработен от Touretzky (2017), надгражда идеите на Kodu Game Lab, като ги адаптира за програмиране на реални роботи. Тази трансформация от виртуален към реален свят дава възможност за прилагане на придобитите знания в по-широк контекст и стимулира интереса на учениците към роботиката и инженерството.

В настоящата работа е направено проучване на възможността за употреба на Kodu като средство за ранно обучение в сферата не само на процедурното и обектно ориентираното програмиране, но и към изучаване на изкуствен интелект. Фокусът е поставен върху принципите на изпълнение на програми в Kodu и тяхното значение за усвояването на логика, вземане на решения, изграждане на йерархии от правила и управление на конфликти. Извършена е съпоставка между Kodu и други платформи за визуално програмиране като MakeBlock и Scratch, както и с езика за правила SWRL. Ранната подготовка не само за работа с изкуствен интелект, но и в посока към осъзнаването как той работи, не е просто въпрос на технологична грамотност, а развива логическо мислене, умения за решаване на проблеми и разбиране на основни концепции, като вземане на решения, анализ на данни и работа с бази от знания. Тези умения е много вероятно да се окажат изключително полезни за бъдещите поколения (Su and Zhong, 2022), когато се очаква интелигентните системи да играят все по-голяма роля в ежедневието.

2. Kodu като средство за въведение в програмирането

Блоковите езици като MakeBlock и Scratch са фокусирани основно върху развиването на алгоритмично мислене. При тях програмирането е основно процедурно. Kodu е по-близък по концепция до езиците за логическо програмиране. Това го прави по-подходящ за развиване на логическо мислене и може да послужи като ефективна пропедевтика за изучаване на изкуствен интелект. Съпоставка е представена в табл. 1. Въпреки изтъкнатите разлики Kodu може да бъде достатъчно

ефективен и при подготовка за работа с процедурни езици за програмиране.

Таблица 1. Съпоставяне на Kodu и MakeBlock/Scratch

Характеристика	**Kodu**	**MakeBlock/Scratch**
Тип на програмиране	Визуално програмиране; Логическо програмиране чрез правила „ако – тогава“	Визуално програмиране; Процедурно програмиране
Представяне на логиката	Визуално дефиниране на правила с WHEN (условие) и DO (действие)	Визуално програмиране чрез блокове за операции
Последователност	Изпълняване на действия според дефинирани правила	Явна последователност от стъпки за изпълнение
Контрол на потока	Условията и действията управляват поведението на героите	Цикли, условни оператори и събития управляват програмата
Работа с обекти	Управление на герои и обекти в игрова среда, чието поведение е базирано на зададени правила	Работа със спрайтове и роботи, чието поведение е базирано на зададени алгоритми
Изпълнение на програми	Правилата се изпълняват паралелно, ако условията са удовлетворени	Изпълнява блоковете последователно
Семантика	Логическа: условията водят до действия	Процедурна: действията се извършват стъпка по стъпка
Обратна връзка	Визуализация на поведението на героите в играта	Визуализация чрез движения на спрайтовете/роботи
Пропедевтика към...	Логическо програмиране (например Prolog)	Процедурно програмиране (например C, Python и др.)

3. Използване на Kodu за пропедевтика за изучаване на изкуствен интелект

Всяка програма на Kodu се състои от една или повече страници, които са съставени от правила. Всяко правило има две части – условие (WHEN) и действие (DO). Програмата в Kodu е изградена около идеи за задаване на условия и действия, които се изпълняват в зависимост от определени ситуации. Това е основа за разбирането на причинно-следствени връзки, което е полезно за развитието на аналитични умения в програмирането и решаването на проблеми. Изпълнението на програмния код в Kodu се подчинява на следните принципи (Touretzky et al., 2016).

L1. Всяко правило се отнася за най-близкия съответстващ обект. В примера на фиг. 1 обектът *kodu* ще се приближи до най-близката ябълка.

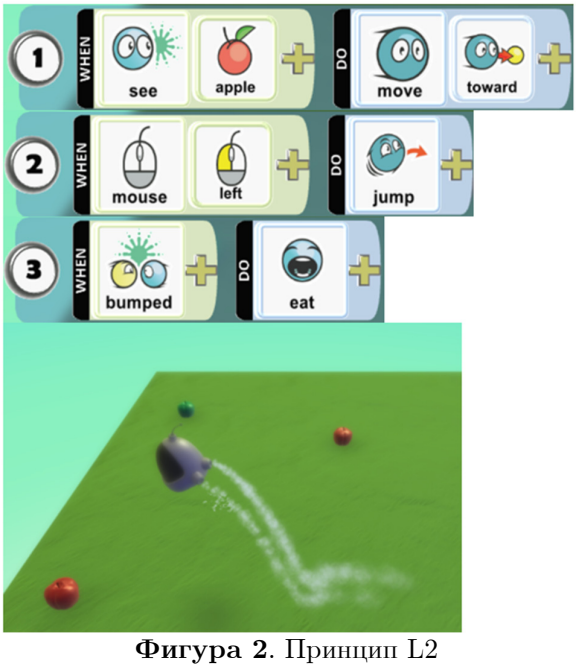
Фигура 1. Принцип L1 за най-близкия съответстващ обект

Чрез това правило учениците се запознават с концепцията за контекстуална логика. Изборът на действие зависи от обекта, който е най-подходящ в дадена ситуация. По подобен начин системите за изкуствен интелект анализират средата и правят избори въз основа на наличните данни.

L2. Когато са налични няколко последователни WHEN условия, техните DO действия ще бъдат изпълнявани паралелно. Например, ако

условията от фиг. 2 на правило 1 и 2 са изпълнени, обектът едновременно ще се придвижва към ябълката и ще скача.

Тази характеристика отделя Kodu от типичния процедурен стил на решаване на сложни задачи чрез линейна поредица от прости стъпки и насочва мисленето на децата в посока на решаване на задачи чрез принципа „разделяй и владей“. Подобни елементи има и в средите за блоково програмиране като Scratch, където всеки отделен герой (sprite) може да се приеме като относително независима изпълнявана паралелно с останалите герои подпрограма. При Kodu този принцип е изведен до крайност, при което разпаралеляването на процесите е налично на ниво алгоритъм за всеки отделен герой.



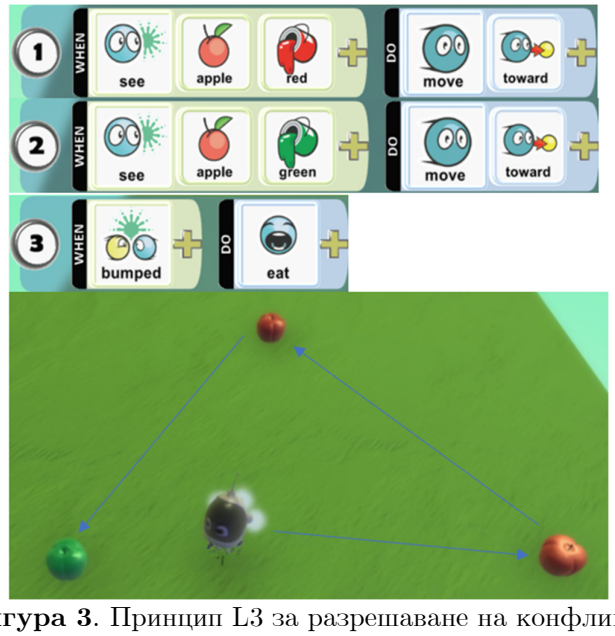
Фигура 2. Принцип L2

L3. Ако две DO действия на правила са в конфликт, ще се изпълни само първото. Условията на правило 1 и правило 2, показани на фиг. 3, са удовлетворени, но действията им не могат да се изпълнят едновременно, затова се изпълнява само първо действие. Така обектът

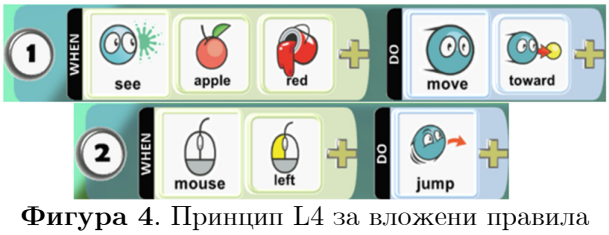
достига и изяжда червената ябълка, въпреки че зелената е по-близо до него.

Принцип L3 съответства на идеята за задаване на приоритети с цел разрешаване на конфликти. Този подход е често срещан в системите за изкуствен интелект, където се налага работа с множество условия, конкуриращи се действия или противоречиви правила.

L4. Едно вложено правило може да се изпълни само ако неговото родителско правило го позволява. Правило 2 на фиг. 4 е вложено, затова обектът *kodu* ще скача при натискане с ляв бутон на мишката, но само когато се движи към червена ябълка.



Фигура 3. Принцип L3 за разрешаване на конфликти



Фигура 4. Принцип L4 за вложени правила

Принцип L4 съответства на йерархичните модели на обучение (дърво на решения, невронна мрежа) за получаване на изходи (действия) в зависимост от предварителни условия, изпълнени в предишни нива на йерархията.

Възможността Kodu да се използва като пропедевтика за изучаване на изкуствен интелект, може да бъде демонстрирана чрез сходствата му със Semantic Web Rule Language (SWRL). SWRL е език за дефиниране на логически правила върху онтологии, което е част от концепцията за семантичен уеб и интелигентни системи. Това включва дефинирането на зависимости и изводи въз основа на съществуващите факти. По подобен начин Kodu дефинира правила за поведение на обекти в играта (табл. 2).

Таблица 2. Съпоставка между Kodu и SWRL

Характеристика	**Kodu**	**SWRL**
Тип на логиката	Визуално дефинирана логика чрез правила if – then	Логически изрази if – then с предикати и условия
Представяне на правила	Визуално, чрез WHEN (условие) и DO (действие)	Текстово, чрез предикати и изрази от типа „→“
Работа с обекти	Действията са свързани с конкретни обекти в играта	Предикатите и правилата се отнасят за обекти в онтология
Семантика	Обектите реагират с действия в съответствие с правила	Декларативно (системата прави изводи на база правила)
Конфликт на правила	Ако две действия са в конфликт, се изпълнява първото поред	Всички правила, чиито условия са верни, се изпълняват
Входни данни	Входът е състоянието на обектите и тяхната среда	Входът са фактите в онтологията
Резултат от правило	Промяна в поведението на обекти в играта	Нови факти, добавени към базата знания

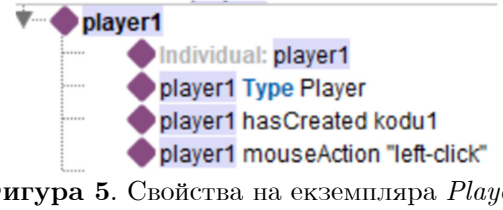
Обратна връзка	Наблюдение на действията на героите в играта	Генериране на нови факти във вид на свойства на обекти в онтологията
Простота на използване	Подходящо за начинаещи, лесно визуално управление	Изисква задълбочено разбиране на логическо програмиране и онтологии

Както Kodu, така и SWRL използват разклонени процеси, работят с обекти или концепции и дефинират правила за управление на техните свойства или поведение. Разликите са в нивото на сложност и начина на приложение: Kodu е визуално ориентиран и достъпен за напълно начинаещи, докато SWRL е текстово базиран и изисква познания по логическо програмиране и семантични технологии.

Аналогията между правилата в Kodu и SWRL може да бъде използвана като пример за това как Kodu може да подготви обучаемите за изкуствения интелект. Чрез разбирането на простите логически правила и структури в Kodu учениците могат да изградят основите на логическото мислене и семантичните правила, които са ключови за разработването на интелигентните системи в изкуствения интелект. По този начин Kodu не само въвежда в концепцията на програмирането, но и подготвя за по-сложни приложения в интелигентните технологии.

За илюстрация на тази възможност създаваме свободно достъпна онтология *kodu*[3] със следните класове (фиг. 6):

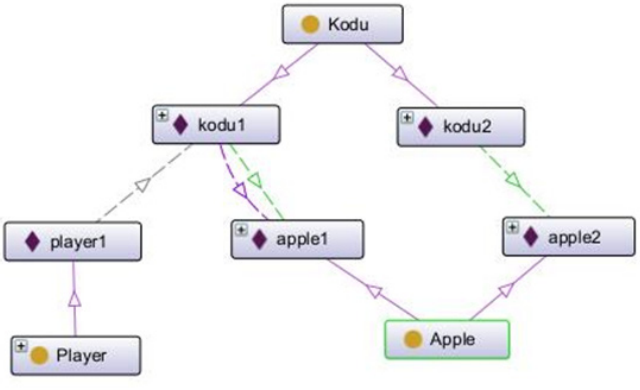
- *Character* с подкласове *Apple* и *Kodu*. Екземплярите от този клас имат свойства color, екземплярите от класа *Kodu* имат свойство *sees*, което показва дали героят вижда друг обект.
- *Player*. Създаден е един екземпляр *Player1* със свойства, показани на фиг. 5. Реализацията е направена в средата на Protégé[4].



Фигура 5. Свойства на екземпляра *Player1*

Фактите в онтологията във вид на триплети са:

```
apple1 Type Apple.
apple1 color "red".
apple2 Type Apple.
apple2 color "green".
kodu1 Type Kodu.
kodu1 sees apple1.
kodu1 Type Kodu.
kodu2 sees apple2.
player1 Type Player.
player1 hasCreated kodu1.
player1       mouseAction
"left-click"
```



Фигура 6. Онтология *kodu*

Изпълняваме следните SWRL правила:

SW1. Ако обект *kodu* вижда червена ябълка, да се придвижи към нея (аналогично на правило 1 от фигура 2 в Kodu).

```
Kodu(?x) ^ sees(?x, ?y) ^ Apple(?y) ^ color(?y, "red") ->
movesTowards(?x, ?y)
```

SW2. Ако играч е създал обект *kodu* и кликва с ляв бутон на мишката, обектът *kodu* придобива способност да скача (аналогично на правило 2 от фигура 2 в Kodu).

```
Player(?p) ^ Kodu(?k) ^ hasCreated(?p, ?k) ^
mouseAction(?p, "left-click") -> jump(?k, true)
```

Резултатът от изпълнението на правилата е добавяне на новите факти:

```
kodu1 movesTowards apple1.
kodu1 jump true.
```

Резултатите от изпълнението на SWRL правилата са нови факти, които се добавят в онтологията. Тези факти могат да бъдат нови свойства на екземпляри, нови връзки между обекти или нова принадлежност към клас. Изпълнението на правилата води до логическо обогатяване на онтологията и извеждане на нови възможности за анализ на съществуващите данни. В конкретния пример според правило SW1 може да се направи заключение, че *kodu1* има свойството (предикат) *movesTowards* със стойност обекта *apple1*; според правило SW2 може да се направи аналогично заключение за свойството *jump* със стойност *true*.

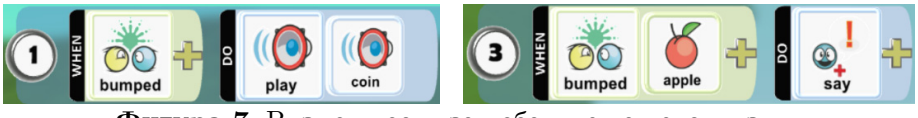
Чрез Kodu деца или начинаещи програмисти могат да се запознаят с концепцията за правила, което е основа за по-сложни концепции в машинното обучение на изкуствени невронни мрежи, където алгоритми вземат решения въз основа на входни данни и правила за изводи, и на интелигентни агенти, които вземат решения въз основа на факти и правила. Дефинираните в Kodu правила са предназначени за взаимодействие с обекти, което може да се разглежда като процес на моделиране на поведение – как героите се движат или реагират на различни събития. Тези умения са важни за разбирането на поведение на агенти в изкуствения интелект, където агенти могат да реагират на различни условия и променящи се фактори.

4. Потенциал за използване на Kodu в българските училища

В обучението по компютърно моделиране в 3. и 4. клас учениците поетапно усвояват основите на алгоритмичното мислене, визуалното програмиране и създаването на дигитални проекти. Чрез игрови и практически дейности те развиват логика, дигитална култура и умения за работа в среда за програмиране. Kodu Game Lab е особено подходящ инструмент за реализиране на тези цели, тъй като предлага интуитивна

визуална среда, в която програмирането се извършва чрез лесно разбираеми блокове и действия, подходящи за възрастта на учениците.

В трети клас учениците се запознават с основни понятия като *алгоритъм* и *цикъл* и се учат да подреждат команди в подходяща последователност. В рамките на визуална програмна среда учениците научават как да разпознават и използват различните елементи на интерфейса – менюта, бутони, сцени и герои. В Kodu те могат да управляват визуално обекти на сцената, да създават герои и сцени, да променят външния им вид и да реализират прости програмни алгоритми. Средата позволява работа с текст и звук, а синхронизирането на действия между героите подпомага разбирането на основите на анимацията и интерактивността (фиг. 7).



Фигура 7. Възможности за добавяне на текст и звук

В четвърти клас обучението се надгражда с въвеждането на по-сложни логически конструкции и нови програмни понятия като *разклонен алгоритъм* и *променлива*. Учениците управляват появата на герои при настъпване на събития и използват множество страници, за да структурират поведението на обекти. Чрез задаването и промяната на точки (score) в игрите те се запознават с концепцията за стойности, които се изменят по време на изпълнение на програмата. Тази възможност може да послужи като подготовка за изучаването на променливи в по-горните класове (фиг. 8). Събитийното програмиране в Kodu прави процеса интуитивен, а създаването на образователни игри – достъпно и мотивиращо, в пълно съответствие с учебните цели.

Фигура 8. Задаване на стойност – белите точки се увеличават с 1

В края на всяка учебна година учениците разработват цялостен дигитален проект – игра, история или анимирана картичка. Kodu е особено подходяща среда за това, тъй като предлага опростен интерфейс, ориентиран изцяло към работа с мишка, което я прави удобна за ученици, които все още развиват уменията си за писане с клавиатура. Програмните конструкции се избират от контекстни менюта, което улеснява разбирането и прилагането на понятия като *условие*, *повторение* и *събитие*.

Въпреки предимствата си в началния етап езикът Kodu по-скоро не е приложим при преподаване на учебното съдържание в 5. клас, защото там вече се въвеждат по-абстрактни и алгоритмично ориентирани структури, като *подпрограми*, *функции*, *структурирано чертане* и *сортиране на данни*. Средата не поддържа създаване на собствени блокове, работа с координатна система или реализация на алгоритми за сравнение и подреждане, което ограничава възможностите ѝ на този образователен етап. В 6. клас фокусът в обучението преминава към програмиране със скриптов текстов език, което изисква по-универсални и функционално разширени програмни среди. Това изключва Kodu като подходяща платформа за този етап на компютърното моделиране.

Въз основа на анализа на възможностите на Kodu и представените в пособието от Георгиева-Трифонова (2018) примери може да бъде разработен и приложен методически инструментариум (учебна програма, списък с учебни задачи, задачи за домашна работа) за използването на Kodu в часовете по компютърно моделиране в 3. и 4. клас като средство за въвеждане в концепциите на изкуствения интелект.

5. Заключение

Kodu предлага възможности за запознаване с ключови концепции в изкуствения интелект на базата на интуитивен визуален подход. Чрез неговите правила и логика учениците могат придобиват основополагащи знания, които могат да бъдат разширени в по-сложни среди за програмиране. Тъй като Kodu използва визуален интерфейс, той е лесен за възприемане от начинаещи, като постепенно ги въвежда в абстракциите на логическо програмиране. След като се запознаят с

принципите на логическите оператори и разклонените процеси, правилата и извеждането на изводи, те могат по-лесно да преминат към по-сложни концепции в SWRL и други логически езици за програмиране, които се използват при изкуствения интелект. Условията и зависимостите в Kodu помагат да се осъзнае как различни входове водят до специфични изходи, което е съществено за обучението на модели в машинното обучение.

БЕЛЕЖКИ

1. Lakelands Academy & Mr T Purslow, Introduction to programming with Kodu, 2020, available at: http://www.lakelandscomputing.com/intro-to-programming.html.
2. A. Angelov, KODU за България, 2012, available at: https://kodu.innovateconsult.net/kodu/
3. T. Georgieva-Trifonova, Kodu ontology, 2024, available at: https://github.com/tsvetankageorgieva/kodu
4. Protégé, available at: https://protege.stanford.edu/

ЛИТЕРАТУРА

Ангелов, А., Момчева, Г. (2012). Технологии и средства за създаване на компютърни игри в класната стая. *Трета международна научно-практическа конференция „Нови технологии в съвременното училище“*, Русе, 23 – 25, ISBN: 978-954-712-577-3.

Ангелов, А., Момчева, Г., Сребрева, Т. (2012a). Визуално програмиране с KODU. *Трета международна научно-практическа конференция „Нови технологии в съвременното училище“*, Русе, 26 – 28 , ISBN: 978-954-712-577-3.

Ангелов, А., Сребрева, Т., Миленкова, Д. (2012b). Визуално програмиране с KODU GameLab. Място и роля на компютърните игри в класната стая. *Научно-практически форум „Иновации в обучението и познавателното развитие“*, Бургас.

Георгиева-Трифонова, Цв. (2018). *Въведение в Kodu.* Издателство „Буквите“, ISBN: 978-619-154-305-2.

Петров, Ф. (2021). *Предизвикателствата пред обучението по информатика в българските средни училища.* Университетско

издателство „Св. Климент Охридски“, София, ISBN:978-954-07-5274-7.

REFERENCES

Angelov, A., Momcheva, G. (2012). Tehnologii i sredstva za sazdavane na kompyutarni igri v klasnata staya. *Treta mezhdunarodna nauchno-prakticheska konferentsia „Novi tehnologii v savremennoto uchilishte“*, Ruse, 23 – 25, ISBN: 978-954-712-577-3. [In Bulgarian]

Angellov, A., Momcheva, G., Srebreva, T. (2012a). Spetsifiki na visualno programirane s KODU. *Treta mezhdunarodna nauchno-prakticheska konferentsiya „Novi tekhnologii v savremenoto uchilishte“*, Ruse, ISBN: 978-954-712-577-3. [In Bulgarian]

Angellov, A., Srebreva, T., Milenkova, D. (2012b). Visualno programirane s KODU GameLab. Myasto i rolya na kompyutarnite igri v klasnata staya. *Nauchno-prakticheski forum „Innovatsii v obuchenieto i poznavatelnoto razvitie“*, Burgas. [In Bulgarian]

Chotova, G. (2021). Application of the Objects-Early Approach to the School Course on Computer Science. *Mathematics, Computer Science and Education*, *4*(1), 7 – 13. DOI: 10.54664/ZCTU5237.

Coy, S. (2013). Kodu game lab, a few lessons learned, *XRDS: Crossroads, The ACM Magazine for Students*, *19*(4), 44 – 47. DOI: 10.1145/2460436.2460450.

Georgieva-Trifonova, Tsv. (2018). *Introduction to Kodu.* Izdatelstvo “Bukvite”, ISBN: 978-619-154-305-2. [In Bulgarian]

Kelly, J. F. (2013). *Kodu for Kids: The Official Guide to Making Your Own Video Games: Create Your Own Video Games for Xbox and PC.* Pearson Education.

Maclaurin, M. (2009). Kodu: end-user programming and design for games. *In Proceedings of the 4th International Conference on Foundations of Digital Games*, (2). DOI: 10.1145/1536513.1536516.

Marghitu, D., Coy, S. (2015). Robotics rule-based formalism to specify behaviors in a visual programming environment. *In Proceedings of the IEEE Blocks and Beyond Workshop*, 45 – 47. DOI: 10.1109/BLOCKS.2015.736899.

Petrov, Ph. (2021). *Predizvikatelstvata pred obuchenieto po informatika v balgarskite sredni uchilishta.* University press "St. Kliment Ohridski", ISBN:978-954-07-5274-7. [In Bulgarian]

Su, J., Zhong, Y. (2022). Artificial Intelligence (AI) in early childhood education: Curriculum design and future directions. *Computers and Education: Artificial Intelligence*, *3*. DOI: 10.1016/j.caeai.2022.100072.

Touretzky, D. S. (2017). Computational thinking and mental models: From kodu to calypso. *Proceedings of the IEEE Blocks and beyond Workshop*, 71 – 78, DOI: 10.1109/BLOCKS.2017.8120416.

Touretzky, D. S., Gardner-Mccune, C., Aggarwal, A. (2016). Teaching "Lawfulness" with Kodu. *Proceedings of the 47th ACM Technical Symposium*, 621 – 626. DOI: 10.1145/2839509.2844652.

THEORETICAL ANALYSIS OF THE POSSIBILITIES OF USING VISUAL PROGRAMMING WITH KODU IN COMPUTER MODELING CLASSES FOR 3RD AND 4TH GRADE

Abstract. The study explores the possibilities of using the visual programming environment Kodu as an alternative to established block programming environments, offering propaedeutics to the youngest students. Within the context of Bulgarian curricula, the potential of using visual programming with Kodu as an effective tool for teaching Computer Modeling in the 3rd and 4th grade is confirmed. An analysis of the Semantic Web Rule Language (SWRL) is conducted, and an illustrative example of an ontology in Kodu is represented. The study shows that the visual programming rules in Kodu can be compared to SWRL rules. This conclusion provides valuable analytical opportunities to extend teaching practices into the field of artificial intelligence. In this way, the Kodu environment ensures a certain advantage because it can be used not only for the first steps in programming, but it can also serve as a propaedeutic for future training in working with artificial intelligence.

Keywords: visual programming; Kodu; artificial intelligence

✉ **DSc. Tsvetanka Georgieva-Trifonova, Prof.**
ORCID iD: 0000-0002-5997-2344
University of Veliko Tarnovo
Veliko Tarnovo, Bulgaria
E-mail: cv.georgieva@live.uni-vt.bg

✉ **Petya Grigorova-Kalcheva**
ORCID iD: 0009-0001-3318-7786
Sofia University "St. Kliment Ohridski"
Sofia (Bulgaria)
E-mail: pbgrigorova@fmi.uni-sofia.bg