Физиката като двигател

<Автор> Михаела Ирина Гюргея
<Автор> Корина Лавиния Тома

<Инфо>

<Ключови думи> анимация, спрайт, блокове, контури, графики, гравитационни закони, сблъсъци, свободно падане, траектория на хвърляне, сила на триене, движение, импулс, оператори, променливи

<Дисциплини> компютърни науки, физика, математика, ICT

<Възраст на учениците> 14–16

<Хардуер> компютър

<Език> Scratch[1]

<Ниво на програмиране> начално, средно

<Резюме>

Какво бихте си помислили, ако ви кажем, че вашите ученици могат да учат по-лесно и едновременно два, на пръв поглед много различни предмета – физика и компютърни науки? В тази глава „двигателят“, т.е. средата за програмиране Scratch[1], е вълшебният инструмент, който ще помогне на учениците да създадат интересни приложения, свързани с ежедневните природни явления, за да разберат по-добре законите на физиката и в хода на всичко това да подобрят уменията си за програмиране.

<Въведение>

Защо използвахме Scratch[1]? Scratch е среда за визуално програмиране, която анимира спрайтове (герои), използвайки блокове на компютърния екран и помага на учениците да създават приложения по-лесно, отколкото с класически програмни среди (C ++, Java и др.). В допълнение, нашият „двигател“ ни помага да преподаваме два предмета, които учениците считат за трудни: физика и компютърни науки. Премахвайки основните препятствия, невъзможността да си представим как действително работи дадено явление и сложният синтаксис на програмирането, ние създадохме приятен нов начин на преподаване.

Учениците, които участваха в разработването на този учебен модул, имаха известен опит в програмирането, така че успяха да се справят със Scratch. Те го използваха, както в редовните, така и в избираеми часове по компютърни науки. Освен това, необходимите им знания по физика са част от стандартната учебна програма и винаги е полезно да се преговори и приложи това, което е научено.

<Какво правят учениците/учителите>

В тази част се редуват последователно обучения по теми, включващи програмиране и физика.

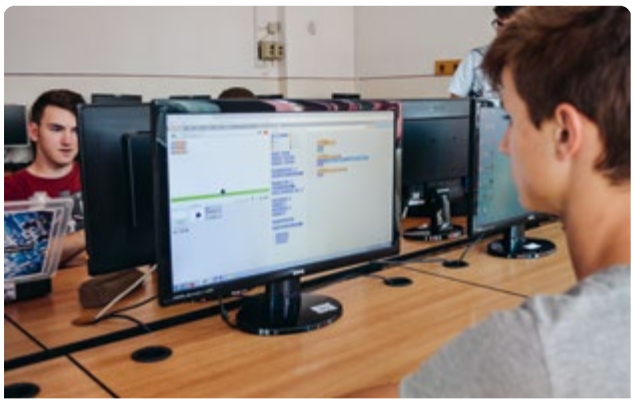
Първо учителят по компютърни науки представя основите за създаване на проект в Scratch[1]. Учениците се запознават с няколко ключови думи, свързани със средата за програмиране Scratch: сцена, спрайтове, костюми и движение. Можете да следвате инструкциите и обясненията на първото приложение за обучение в Scratch, което е без формули от физиката.[2]

Учениците трябва да разберат взаимодействията между спрайтовете и тяхната синхронизация, както и как работи координатната система. Можете да намерите пълен урок за Scratch онлайн.[3]

За да разберат по-добре основните алгоритми в Scratch, учениците разгледаха вълнуващи и интересни приложения на сайта на Scratch. Виждайки кода зад тях, бяха изненадани да открият, че сами могат да създават такива приложения.

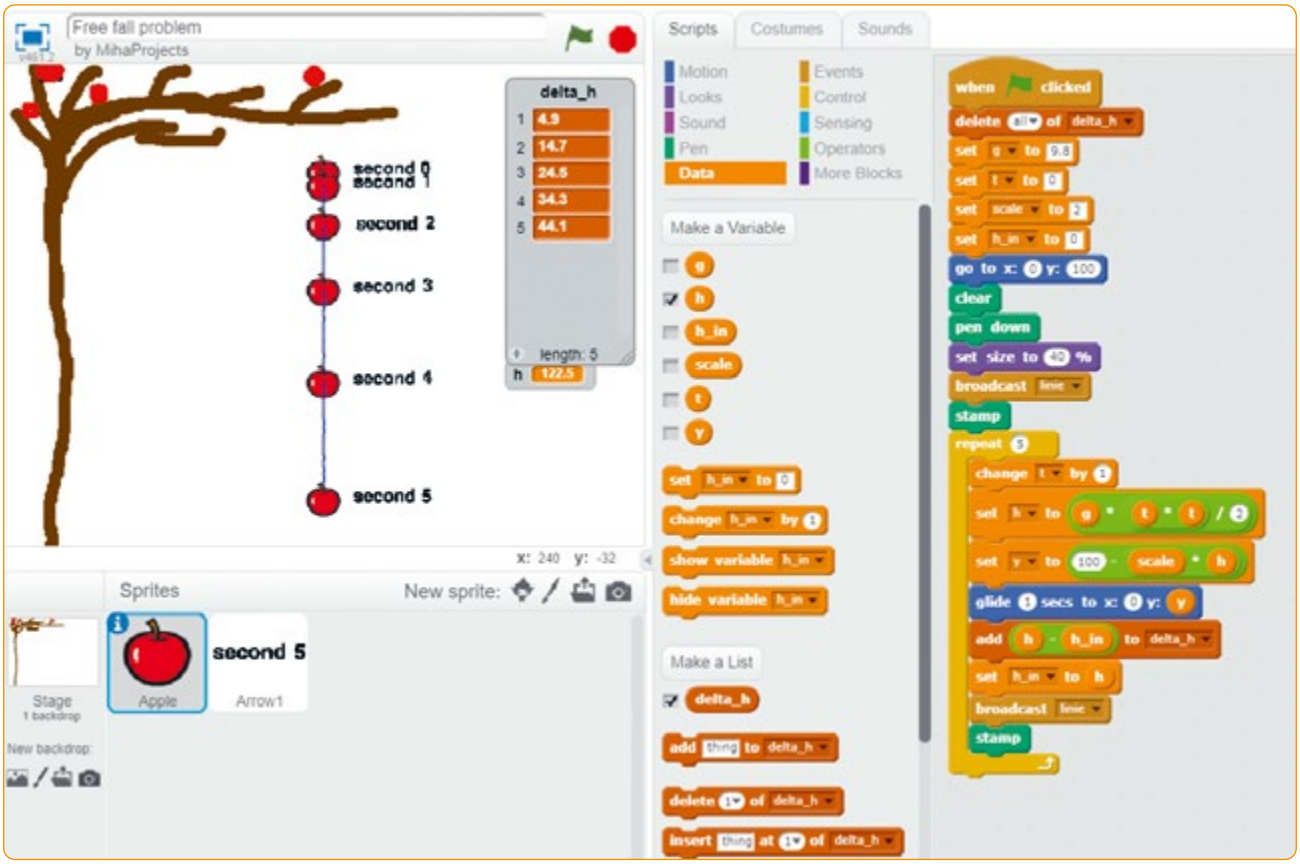
За частта по физика учениците прилагаха теоретичните принципи, стоящи зад явленията от света около тях. Учителят по физика предложи голямо разнообразие от теми[4], които учениците след това обсъдиха: необходими формули, възможна анимация, дизайн и т.н.

Учениците избраха някои от тези теми. Седмица по-късно получихме приложения за траекторията на хвърляне на топка, свободното падане на ябълка, по-сложното падане на капка вода или сблъсък между две топки, а също и за движението на планетите в Слънчевата система или дори малки игри.(📷1)



📷 1: Индивидуална работа

Накрая всички представиха проекта си пред класа и получиха обратна връзка от своите връстници. Това



2: Задача свободно падане

улесни учениците да разберат кои части от проектите им трябва да бъдат подобрени: програмирането или физиката.

В края на нашия проект по-големите ученици станаха учители на по-малките ученици (12–13 години), като представиха подходящи приложения и ги тестваха по време на уроците по физика. Те се радваха на внимание и много се гордееха с работата си. По-големите ученици също получиха предложения от по-малките ученици. Най-добрите от тези симулации за ученици е достъпно в платформата Scratch.[2]

Следващите раздели съдържат примери как подходихме към частта по физика и програмиране.

<Приложение 1: Проблем със свободно падане (ябълка на Нютон)>

Всеки ученик е чувал за свободното падане на този исторически обект – ябълката на Нютон.

Приложението на 2 е вдъхновено от класически проблем. Какво е разстоянието, изминато всяка секунда от ябълката на Нютон при свободно падане?

Физика теория

Ние намираме линейното движение с постоянно гравитационно ускорение $g = 9{,}8\ \frac{\text{m}}{\text{s}^2}$.

След време t изминатото разстояние $h(t)$ на ябълката е: $h(t) = \frac{gt^2}{2}$.

Началната точка $h(0)$ е фиксирана върху клона на дървото, от който ябълката се отделя.

След това изчисляваме изминатото разстояние за по-дълъг период, $t + \Delta t$: $h(t+\Delta t) = \frac{g(t+\Delta t)^2}{2}$.

Общата формула за разстоянието $\Delta h(t)$, разстоянието, изминато от ябълката за времето Δt, е:

$$\Delta h(t) = h(t+\Delta t) - h(t) = \frac{g(2t\Delta t + \Delta t^2)}{2}.$$

Ще използваме данните за конкретния проблем, за да персонализираме основната формула; в този случай 1сек за Δt. За първата секунда $t = t_{in} = 0$ резултатът е $\Delta h_1 = 4{,}9$ м, за следващата секунда, t=1сек резултатът $\Delta h_2 = 3 \cdot 4{,}9$ м = 14,7 м и т.н. Чрез математическата логика можем да изчислим изминатото разстояние през n-тата секунда, отчитайки $t = (n-1)$ сек:

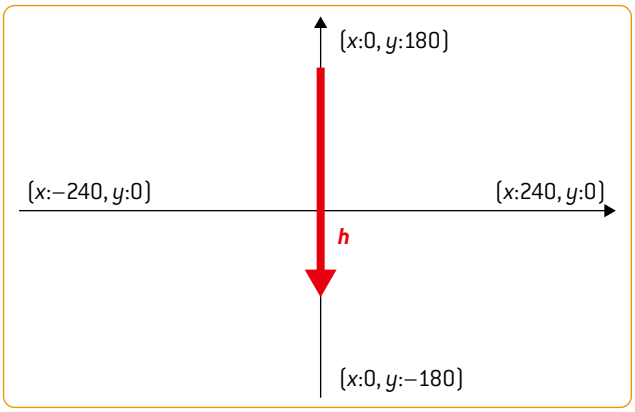
$$\Delta h_n = \frac{9{,}8(2n-1)}{2}\ \text{m}.$$

Тогава учениците могат да изчислят, а също и да видят в нашата анимация, че изминатото разстояние се увеличава със същата стойност всяка секунда – 9,8 м.

Как да програмираме това?

Използвани променливи:
g: гравитационно ускорение
t: брояч на секунди (със стойности: 0, 1, 2, 3, 4, 5)
h: изминатото разстояние след *t* секунди
h_in: първоначалната позиция на ябълката
delta_h: списък (масив) с всички изминати разстояния във всяка секунда
y: y-координата на ябълката
Забележка: В това приложение координатата x остава постоянна = 0, така че можете да преместите по-лесно траекторията наляво или надясно на сцената.

В началото ябълката е в точката с координатите (0,180). Началната точка и посоката за изминатото разстояние *h*(*t*) са отбелязани на 3.



3: Ориентация в координатна система

```
Free_fall

delta_h: array of real
g←9.8
t←0
h_in←0
scale←2
go to (0,100)
clear()
pen(down)
stamp()

while (t<5)
    t← t + 1
    h← g * t * t / 2
    y←100-scale*h
    glide (0, y)
    add (delta_h, h-h_in)
    h_in ←h
    broadcast(linie)
    stamp()
```

4: Задача свободно падане

Използвайки цикъла 5 пъти, преизчислихме разстоянието, което ябълката изминава след всяка секунда, изчислявайки новата y-координата и отчитайки характеристиките на екрана. Вижте 4 за алгоритъма на програмата.

Предизвикателство

Учениците модифицират Δ*t* и изминатото време *t* (ябълковото ни дърво може да е много високо – вероятно е по-добре да нарисуваме кула) или преместват задачата върху друга планета със собствено гравитационно ускорение. За да създадат сложен проект, те биха могли да добавят силата на триене от въздуха и да обмислят променливо гравитационно ускорение, като хвърлят ябълката от метеорологичен балон на по-голяма надморска височина.

<Приложение 2: Падаща капка вода>

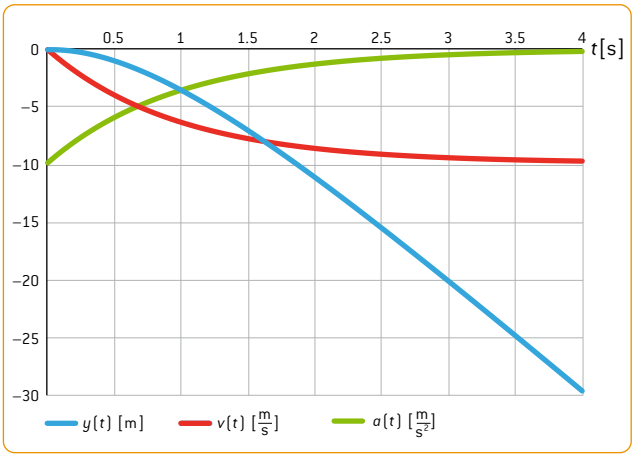
В дъждовен ден всеки може да наблюдава падането на водни капки. Учениците анализираха линейното движение на една капка с нашата симулация. В началото те видяха, че падането на водната капка се ускорява, но с намаляващо ускорение. След известно време скоростта на капката достигна своята граница, крайната скорост v_t, когато ускорението достига нула. След това капката вода продължи да се движи с тази постоянна скорост. Как можете да обясните това?

Физика теория

В ускоряващата част от движението, две сили в противоположни посоки действат на капката: гравитационната сила $G = mg$ (*m*: маса на капката, *g*: гравитационно ускорение) и силата на триене $F_f = kv$ (*k*: константа на пропорционалност, *v*: моментна скорост). Ускорението на капката става: $a = g - \frac{k}{m} v$.

В нашата симулация измерваме голяма капка, с диаметър около 5 мм с крайна скорост $v_t = 9{,}8\ \frac{m}{s}$.[5] В този случай константата е $\frac{k}{m} = \frac{1}{s}$. Ускорението намалява, когато скоростта се увеличава (виж 5). Първоначалните стойности са $a = 9{,}8\ \frac{m}{s^2}$, $v = 0$ и $y = 0$.

Използваме малка програма в C++[4], за да изчислим моментното ускорение и скорост, където отчитаме ускорението и константата на скоростта за много малки времеви интервали Δ*t* (като 0,05 сек). В този случай скоростта се увеличава $\Delta v = a\Delta t$, а изминатото разстояние с $\Delta y = v\Delta t$ за всеки избран Δ*t* (метод стъпка по стъпка).

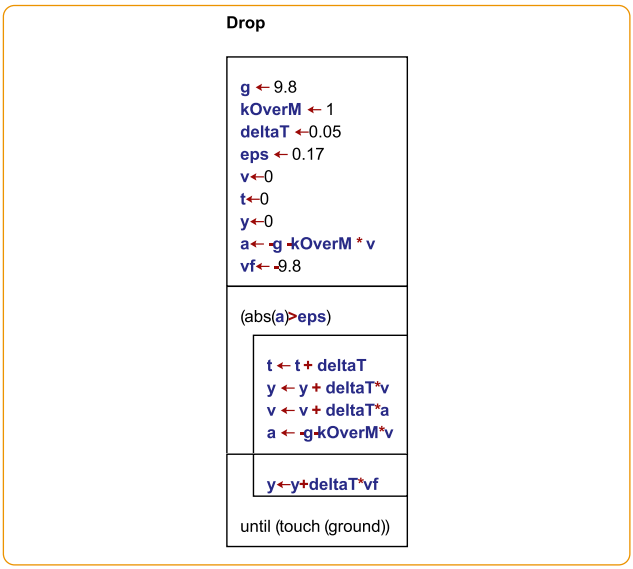


5: Отношение между изминатото разстояние, скоростта и ускорението

Как да програмираме всичко това?

1. Оцветете спрайта водна капка.
2. Оцветете една хоризонтална зелена линия в долната част на фона.
3. Направете спрайт със съобщението „ускорение = 0“, което се появява, когато ускорението има стойност около 0.
4. Напишете кода за спрайта – капка. Капката старира в точката $(0, y_{init})$. Използвайки цикъл, преизчисляваме $a(t)$, $v(t)$, $y(t)$. Използваме разстоянието, достигнато при падането след всеки Δt, като изчисляваме новата y-координата и отчитаме характеристиките на екрана. Ускорението намалява и когато е около 0, цикълът е завършен. В този момент на екрана се показва спрайта със съобщението. След това капката пада с постоянна скорост, докато не докосне зелената линия на фона.

6 дава ясно обяснение на кода.



6: Падаща капка

Предизвикателство

Учениците могат да подобрят това приложение, ако добавят променлива за масата на водната капка (диаметърът на водната капка обикновено може да приеме стойности от 1 мм до 5 мм)[6] и друг вид сила – на триене: $F_f = \frac{kv^2}{2}$.

Учениците могат също да добавят повече капки с различни маси и да сравнят как падат.

<Приложение 3: Еластичен сблъсък>

Има много примери за сблъскване на тела около нас. Тези сблъсъци са сложни, но ние считаме еластичните сблъсъци за приложими в реалния живот за билярдни или стоманени топки или на теория за сблъсъци на молекули, когато учениците изучават модела на идеалния газ.

Физика теория

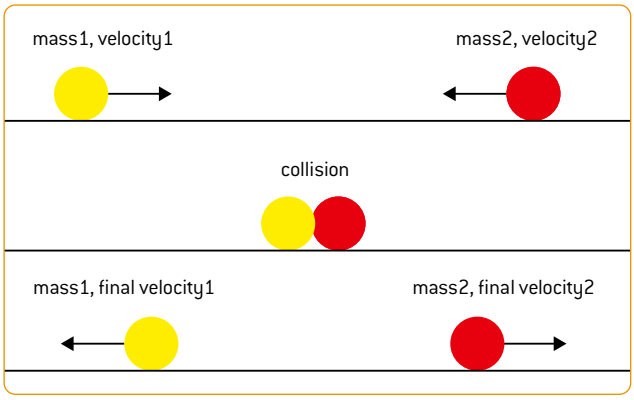
Импулса и кинетичната енергия за две топки с масата m_1 и m_2, начални скорости $(\vec{v_1})$ и $(\vec{v_2})$ и крайни скорости $(\vec{v_{1f}})$ и $(\vec{v_{2f}})$, движещи се една срещу друга, се запазват. [7]

$$m_1\vec{v_1} + m_2\vec{v_2} = m_1\vec{v_{1f}} + m_2\vec{v_{2f}} \text{ и}$$

$$\frac{1}{2}m_1v_1^2 + \frac{1}{2}m_2v_2^2 = \frac{1}{2}m_1v_{1f}^2 + \frac{1}{2}m_2v_{2f}^2$$

Ако движението се извършва по една и съща линия (движение по x-оста), можем да използваме знаци + или -, за да обозначим посоките. Векторното обозначение на скоростта не е необходимо за случая на сблъсък по права линия и крайните скорости могат да бъдат изчислени по следните уравнения:

$$v_{1f} = 2\frac{m_1v_1 + m_2v_2}{m_1 + m_2} - v_1 \text{ и } v_{2f} = 2\frac{m_1v_1 + m_2v_2}{m_1 + m_2} - v_2.$$



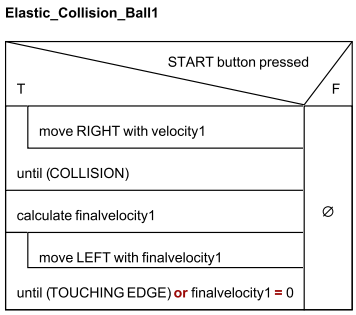
7: Еластичен сблъсък

Как се програмира това?

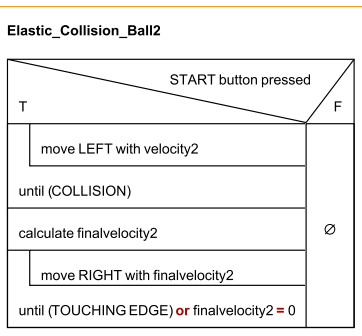
1. Изберете два спрайта за топките (Ball1 и Ball2) и един спрайт за бутона **START** (START спрайт).

2. Използвайте променливи: *маса1*, *маса2*, *скорост1*, *скорост2* (масата и началната скорост) за всеки обект. Направете променливи плъзгачи и задайте минималната и максималната стойност за тях.
3. Въведете масата и началните скорости за всеки обект.
4. Натиснете бутона **START**. В този момент спрайтът изпраща съобщение на спрайтовете на топката. Когато получат съобщението, всяка топка се придвижва към другата, използвайки добре познатата формула *разстояние = скорост × време*.
5. Изчислете крайните скорости на топките и ги използвайте за придвижване на топките в правилната посока, докато топката докосне ръба или напусне сцената, или остане на мястото си, защото новата ѝ скорост е 0.

8 и 9 дават ясна представа за начина, по който са анимирани двете топки в Scratch[1].



8: Еластичен сблъсък за monka 1



9: Еластичен сблъсък за monka 2

Два примера за използване на това приложение:

1. Изберете едната скорост да е равна на 0 и еднаква маса за топките; след този сблъсък ще наблюдавате, че движещата се топка спира, а другата се движи със същата скорост, която първата топка е имала преди удара.
2. Топките имат различна скорост и еднаква маса; след сблъсъка ще видите, че обектите приемат стойността за скорост на другата.

И в двата примера топките променят инерцията си.

Предизвикателство

Учениците могат да променят размера на топките правопропорционално на тяхната маса; те биха могли да направят приложение за двуизмерен еластичен сблъсък (симулация за ефекта на Комптон) или биха могли да програмират симулация за сблъсък на топка със стена. Можете да продължите изследването с друга физична теория, например. нееластичен сблъсък.[4]

<Заключение>

<За учениците>

Предимства

Учениците учат теорията по физика по по-приятен начин и успяват да разберат по-добре природните явления, използвайки симулации в Scratch. Те задълбочават едновременно знанията си по компютърни науки и физика. Въпреки че не всички техни проекти бяха перфектни, учениците определено подобриха уменията си за програмиране и алгоритмично мислене.

Недостатъци

Учениците работят сами и повече вкъщи. Получават обратна връзка в училище.

<За учителите>

Предимства

Наблюдавахме реален интерес на учениците към създаването на оригинално приложение и така научават повече, отколкото в класическите уроци.

Недостатъци

Беше ни трудно да координираме целия клас, поради голямото разнообразие от теми по физика и много грешки във всяко приложение. Смятаме, че би било по-добре да дадем на всички ученици една и съща тема и да ги насърчим да я подобрят максимално, в зависимост от нивото на подготовка и възможностите им.

<Дейност на сътрудничество>

Ученици от различни училища и страни биха могли да решат предизвикателствата в проектите и да създадат нови идеи свързани с първоначалната тема. Всички тези приложения могат да бъдат събрани в платформата Scratch и да се организира конкурс за определяне най-добрите от тях в категории според нивото на програмиране и физика.

<Препратки>

[1] https://scratch.mit.edu/
[2] www.science-on-stage.de/coding-materials.
* Останалите препратки може да видите в английската верисия.

<Печат>

<Публикувано от>
Science on Stage Deutschland e. V.
Ам Борсигтурм 15
13507 Берлин, Германия

<Главен координатор>
↳ **Д-р Йорг Гуцанк**, Гимназия Лайбниц | Международно училище в Дортмунд, Дортмунд, Германия
Председател на Science on Stage Deutschland e. V.

<Координатори>
↳ **Себастиан Фънк**, Вила Виверсбуш, Велберт – Лангенберг, Германия, Член на съвета на Science on Stage Deutschland e. V.
↳ **Жан – Люк Рихтер**, Гимназия Jean-Baptiste Schwilgué, Селестат, Франция, Заместник-председател на Наука на сцената Франция
↳ **Бернар Шриек (ret.)**, Гимназия Marien, Верл, Германия

<Цялостна координация и редактиране>
↳ **Даниела Нюман**, Проектен мениджър на Science on Stage Deutschland e. V.
↳ **Стефани Шлунк**, Изпълнителен мениджър на Science on Stage Deutschland e. V.
↳ **Йохана Шулце**, Заместник изпълнителен мениджър на Science on Stage Deutschland e. V.

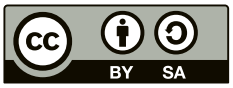
<Корекция и превод>
Мария Петрова, Петър Андреев, Десислава Цокова и Моника Ковачка-Димитрова

<Дизайн>
WEBERSUPIRAN.berlin

<Илюстрации>
Рупърт Таке, TricomKommunikation und Verlag GmbH

<С подкрепата на>
SAP SE

<За поръчка>
www.science-on-stage.bg
office@frgi.bg



Първо издание 2020