**COLEGIUL NAȚIONAL „ GRIGORE MOISIL ”**

**URZICENI**

**JUDEȚUL IALOMIȚA**

**Examen pentru obținerea atestatului**

**profesional în informatică**

**PROIECT** : „Marksman” – Joc interactiv

**NUME** : Simion

**PRENUME** : Cristian

**CLASA** : a XII-a A

**PROF. ÎNDRUMĂTOR** : Bardac Sorin

**Sesiunea Mai 2018**

Doamnă Director,

Subsemnatul Simion Cristian, elev în clasa a XII-a A, filiera teoretică, profil real, specializarea matematică-informatică, intensiv informatică, vă rog să îmi aprobați cererea de înscriere la examenul pentru obținerea atestatului profesional în informatică, sesiunea mai 2018.

Menționez că, la proba de proiect voi susține tema ,,Marksman” - joc interactiv .

Data: 13.12.2017 Semnătura:

Doamnei Director al Colegiului Național ,,Grigore Moisil”

# Colegiul Naţional „Grigore Moisil”

# Urziceni

# Judeţul Ialomiţa

# Catedra de informatică

**Referat**

Lucrarea de față are tema: „Marksman” și este o pagină web creată pentru prezentarea unui joc. Aceasta s-a realizat în timpul orelor de laborator, iar conținutul său prezintă noțiunile din capitolele studiate.

Aplicația a fost realizată în limbajele HTML, CSS și JavaScript prin scrierea de linii de cod.

**Profesor îndrumător : Bardac Sorin**

I. Introducere

„Marksman” este un joc de îndemânare, totodată putând fi considerat un simulator de fizică mecanică. Este destinat persoanelor de orice vârstă deoarece controalele și regulile sunt simple, dar în același timp jocul constituie o provocare. Pentru crearea jocului am creat un site în HyperText Markup Language (HTML), stilul fiind stabilit prin standardul de formatare Cascading Style Sheets (CSS). Jocul în sine a fost programat în limbajul JavaScript (JS). Programarea a fost realizată prin scrierea liniilor de cod în editorul de text Notepad++.

**Reguli:**

* Ai 60 de secunde la dispoziție pentru a obține un scor cât mai mare.
* Poți obține puncte nimerind ținta ce se află în mișcare. Vei primi:
  + **1 punct**: dacă săgeata este în urcare atunci cand se lovește de țintă,
  + **2 puncte**: dacă săgeata este în coborâre atunci cand se lovește de țintă.
* Nu poți pregăti o altă săgeată în timp ce una este deja din aer, trebuie să aștepți până când săgeata iese din marginile orizontale sau se lovește de țintă sau de podea.
* Când timpul ajunge la 0, un text ce indică sfârșitul jocului și scorul final vor apărea, alături de un buton ce repornește jocul atunci când este apăsat click pe el.

**Controale:**

* Jocul este jucat doar cu mouse-ul.
* Apasă click oriunde pe ecranul jocului si ține apăsat pentru a trasa un segment.
* În funcție de lungimea segmentului (până la o lungime maximă) si de unghiul dintre segmentul trasat si axa orizontală, puterea si unghiul de lansare ale săgeții se vor modifica.
* Dă drumul click-ului pentru a lansa săgeata!

II. Structura și conținutul proiectului

Pentru lansarea jocului este nevoie doar de un browser web ce suportă elementul <canvas>. Se deschide fișierul „index.html” (sau oricare dintre fișierele cu extensia .html). Site-ul este realizat din 3 pagini web, denumite: „Acasă” („index.html”), „Instrucțiuni” („instructiuni.html”) și „Joc” („joc.html”). Acestea pot fi accesate de pe oricare pagină din cadrul site-ului, apăsând pe butoanele din cadrul barei de navigație, realizate cu ajutorul hyperlink-urilor. Butoanele au același nume cu cel al paginii web, cu excepția paginii „Joc”, căruia îi corespunde butonul „Joacă!”.

Pentru a proiecta jocul am folosit elementul <canvas> ce permite desenarea diferitelor forme și imagini pe o pânză virtuală. I-am atribuit id-ul „joc” pentru a putea să o manipulez în interiorul JavaScript-ului.

În JavaScript, am declarat variabilele canvas și ctx, în care am salvat proprietățile canvas-ului, și contextul de desenare asociat acelui canvas, în acest caz acesta fiind ‘2d’ deoarece jocul este 2D:

**var** canvas **=** **document.**getElementById**(**"joc"**);**

**var** ctx **=** canvas**.**getContext**(**'2d'**);**

Am redimensionat canvas-ul pentru a fi încadrat corespunzător în pagină, noile valori fiind 70% din înălțimea, respectiv lățimea ecranului.

canvas**.**width **=** **window.innerWidth** **\*** 0.7**;**

canvas**.**height **=** **window.innerHeight** **\*** 0.7**;**

Am declarat variabilele și constantele globale de care voi avea nevoie în realizarea jocului:

**var** v0 **=** 0**,** alfa **=** 0**;**

→ viteza inițială a săgeții și unghiul dintre vectorul de viteză inițială și axa orizontală, necesare pentru calculul traiectoriei.



**var** framesPerSecond **=** 60**,** gameStart **=** **false,** gameOver **=** **false,** gameTime **=** 60**;**

→ cadrele pe secundă, reprezentând de câte ori pe secundă se desenează un nou cadru, două variabile booleene ce indică dacă jocul a început/s-a sfârșit, și o variabilă ce indică timpul rămas în secunde.



**var** mouseInitX**,** mouseInitY**,** mouseFinX**,** mouseFinY**,** mousePressed **=** **false;**

→ coordonatele poziției inițiale, respectiv finale ale mouse-ului, precum și o variabilă booleană ce indică dacă este apăsat click.



**var** arrowInAir **=** **false,** passedMiddle**,** power**,** score **=** 0**;**

→ o variabilă ce indică dacă o săgeată este în aer, o variabilă ce indică dacă săgeata a trecut de jumătatea distanței orizontale totale, folosită la desenarea săgeții și acordarea de puncte bonus, puterea și scorul, ce vor fi afișate pe ecran.



**var** currentTime**,** prevY**,** currentY**,** currentX**,** nextX**;**

→ prima variabilă reține timpul corespunzător săgeții, iar următoarele variabile rețin coordonatele săgeții în 2 momente consecutive.



**var** targetX **=** canvas**.**width **\*** 0.9**,** targetY **=** canvas**.**height **\*** 0.5**,** targetWidth **=** 20**,** targetHeight **=** 80**,**

targetSpeed **=** 1000 **/** framesPerSecond **\*** 0.15**;**

→ coordonatele inițiale, dimensiunile și viteza pe verticală a țintei; viteza depinde de cadrele pe secundă.



**var** toDrawX**,** toDrawY**;**

→ coordonatele punctelor ce vor fi desenate pe traiectoria ajutătoare.



**var** arrowsStuckTarget **=** **[],** arrowsStuckFloor **=** **[];**

→ vectori ce rețin poziția și unghiul sub care săgeata se înfige în obstacol, respectiv podea. Acestea au 3 câmpuri: x, y și angle; x și y reprezentând coordonatele, iar angle unghiul în timpul impactului. În cazul primului vector, coordonata y este relativă la înălțimea țintei

(y ∈ [0, targetHeight]) deoarece ținta se mișcă pe verticală.



**const** pozArcX **=** 50**,** pozArcY **=** canvas**.**height **\*** 0.85**,**

podeaY **=** canvas**.**height **\*** 0.9**;**

→ coordonatele poziției arcului, reprezentând de asemenea punctul de origine al traiectoriei și coordonata podelei.



**const** g **=** 9.80665**,** distMax **=** 150**,** vmax **=** 250**;**

→ accelerația gravitațională, folosită în ecuațiile de determinare a poziției, distanța maximă a segmentului trasat cu mouse-ul, și viteza maximă pe care o poate avea săgeata.

De asemenea, am salvat în variabile imaginile (tile-urile) cu săgeata și arcul în diferitele stagii de tragere. Imaginile se află în dosarul „tiles” din dosarul principal:



**var** arrow **=** **new** Image**();**

arrow**.**src **=** "tiles/arrow.png"**;**

**var** sprite **=** **[];**

**for(var** j **=** 0**;** j **<** 12**;** j**++)** **{**

sprite**[**j**]** **=** **new** Image**();**

**var** cifraZecilor **=** Math**.**floor**(**j **/** 10**);**

**var** cifraUnitatilor **=** j **%** 10**;**

sprite**[**j**].**src **=** "tiles/tile0" **+** cifraZecilor **+** cifraUnitatilor **+** ".png"**;**

**}**

* Atribuirea imaginilor în vectorul sprite este realizată cu ajutorul concatenării, cifrele zecilor și unităților fiind convertite în text.

La încărcarea completă a ferestrei, ecranul de început este desenat:

**window.onload** **=** **function()** **{**

ctx**.**fillStyle **=** 'black'**;**

ctx**.**fillRect**(**0**,** 0**,** canvas**.**width**,** canvas**.**height**);**

ctx**.**fillStyle **=** 'white'**;**

ctx**.**textAlign **=** "center"**;**

ctx**.**font **=** "70px Georgia"**;**

ctx**.**fillText**(**"Marksman"**,** canvas**.**width **\*** 0.5**,** canvas**.**height **\*** 0.4**);**

ctx**.**beginPath**();**

ctx**.**rect**(**canvas**.**width **\*** 0.5 **-** 150**,** canvas**.**height **\*** 0.5 **+** 160 **-** 50**,** 300**,** 100**);**

ctx**.**fill**();**

ctx**.**fillStyle **=** 'black'**;**

ctx**.**font **=** "42px Georgia"**;**

ctx**.**fillText**(**"Începe jocul!"**,** canvas**.**width **\*** 0.5**,** canvas**.**height **\*** 0.5 **+** 170**);**

**}**

Ecranul rezultat arată astfel:



Elementul principal al jocului este lansarea săgeții. Această lansare poate fi văzută drept o **aruncare sub un unghi** din domeniul fizicii mecanice. Pentru a folosi formulele corespunzătoare trebuie să cunoaștem *viteza inițială* (v0), *unghiul* dintre vectorul vitezei inițiale și axa orizontală (α – alfa), *accelerația gravitațională* (g) și *timpul* (t). Astfel, coordonatele x și y se pot calcula folosind următoarele formule:

Se observă că singurul număr care se modifică în timpul unei lansări este timpul, deci putem crea două funcții ce au ca parametru timpul t:

**function** coord\_x**(**t**)** **{**

**return** v0 **\*** t **\*** Math**.**cos**(**alfa**);**

**}**

**function** coord\_y**(**t**)** **{**

**return** v0 **\*** t **\*** Math**.**sin**(**alfa**)** **-** g **/** 2 **\*** t **\*** t**;**

**}**

Am creat de asemenea o funcție ce returnează distanța euclidiană dintre 2 puncte:

**function** euclidianDistance**(**x1**,** y1**,** x2**,** y2**)** **{**

**return** Math**.**sqrt**((**x1 **-** x2**)** **\*** **(**x1 **-** x2**)** **+** **(**y1 **-** y2**)** **\*** **(**y1 **-** y2**));**

**}**

Am adăugat câte un event listener pentru fiecare tip de acțiune a mouse-ului: apăsarea click-ului ('mousedown'), dat drumul click-ului ('mouseup') și mișcarea mouse-ului ('mousemove'):

canvas**.**addEventListener**(**'mousedown'**,**

**function(**evt**)** **{**

**if(!**gameStart **&&** ctx**.**isPointInPath**(**mouseFinX**,** mouseFinY**))** **{**

gameStart **=** **true;**

**setInterval(**draw**,** 1000 **/** framesPerSecond**);**

**}**

**else** **if(**gameOver **&&** ctx**.**isPointInPath**(**mouseFinX**,** mouseFinY**))** **{**

resetGame**();**

**}**

**else** **if(!**arrowInAir **&&** gameStart**)** **{**

**var** mousePos **=** calculateMousePos**(**evt**);**

mouseInitX **=** mousePos**.**x**;**

mouseInitY **=** mousePos**.**y**;**

mousePressed **=** **true;**

**}**

**});**

**Acest event listener are în vedere: pornirea jocului, restartarea jocului și calcularea punctului de început al segmentului.**

Funcția isPointInPath verifică dacă punctul de coordonate x și y - unde x este primul argument și y al doilea - se află în interiorul ultimei forme desenate. În prima structură alternativă aceasta se referă la butonul în formă de dreptunghi din ecranul de început, iar în a doua structură alternativă se referă la butonul în formă de elipsă ce va apărea la sfârșitul jocului.

Metoda setInterval execută procedura din primul câmp de un număr de ori specificat în al doilea câmp în milisecunde. Astfel, pentru a desena jocul de framesPerSecond-ori pe secundă, intervalul dintre cadre trebuie să fie egal cu 1 (o secundă) împărțit la cadrele pe secundă, deci va fi egal cu 1000 (în milisecunde) / framesPerSecond.

Procedura resetGame aduce jocul în stadiul inițial, ștergând săgețile înfipte în podea și în țintă, setând scorul la 0, timpul înapoi la 60 de secunde, și recentrează ținta în poziția inițială.

Funcția calculateMousePos calculează poziția mouse-ului și o returnează în câmpurile x și y.

canvas**.**addEventListener**(**'mouseup'**,**

**function()** **{**

**if(!**arrowInAir **&&** mousePressed **&&** **!**gameOver**)** **{**

mousePressed **=** **false;**

arrowInAir **=** **true;**

passedMiddle **=** **false;**

currentTime **=** 4 **/** framesPerSecond**;**

prevY **=** 0**;**

nextX **=** coord\_x**(**currentTime**);**

**}**

**});**

**Acest event listener are în vedere: lansarea unei săgeți.**

Se inițializează valorile necesare desenării corecte a unei săgeți.

Pentru a simplifica procesul, „timpul” săgeții trece mai rapid decât timpul real (cel de desenare al cadrelor). Coordonata x a săgeții se va calcula apelând funcția coord\_x, având drept argument timpul săgeții (currentTime). Coordonata y anterioară a săgeții este 0.

canvas**.**addEventListener**(**'mousemove'**,**

**function(**evt**)** **{**

**var** mousePos **=** calculateMousePos**(**evt**);**

mouseFinX **=** mousePos**.**x**;**

mouseFinY **=** mousePos**.**y**;**

**});**

**Acest event listener are în vedere: calcularea punctului de sfârșit al segmentului.**

Procedura centrală a programului este draw(). În interiorul procedurii se execută toate celelalte operații de desenare a unui cadru:

La început se desenează fundalul și podeaua.

**function** draw**()** **{**

ctx**.**fillStyle **=** 'grey'**;**

ctx**.**fillRect**(**0**,** 0**,** canvas**.**width**,** canvas**.**height**);**

ctx**.**fillStyle **=** 'black'**;**

ctx**.**fillRect**(**0**,** podeaY **+** 1**,** canvas**.**width**,** 2**);**

Apoi, se verifică dacă timpul s-a scurs. În acest caz, se apelează subprogramul drawGrameOver, ce desenează ecranul de sfârșit de joc.

**if(**gameTime **<=** 0**)** **{**

gameOver **=** **true;**

drawGameOver**();**

**}**

**function** drawGameOver**()** **{**

ctx**.**fillStyle **=** "white"**;**

ctx**.**textAlign **=** "center"**;**

ctx**.**font **=** "70px Georgia"**;**

ctx**.**fillText**(**"Timpul a expirat!"**,** canvas**.**width **\*** 0.5**,** canvas**.**height **\*** 0.35**);**

ctx**.**font **=** "40px Georgia"**;**

ctx**.**fillText**(**"Scorul tău: " **+** score**,** canvas**.**width **\*** 0.5**,** canvas**.**height **\*** 0.35 **+** 80**);**

ctx**.**beginPath**();**

ctx**.**ellipse**(**canvas**.**width **\*** 0.5 **,**canvas**.**height **\*** 0.5 **+** 160**,** 120**,** 50**,** 0**,** 0**,** Math**.**PI **\*** 2**);**

**if(**ctx**.**isPointInPath**(**mouseFinX**,** mouseFinY**))**

ctx**.**fillStyle **=** "white"**;**

**else**

ctx**.**fillStyle **=** "black"**;**

ctx**.**fill**();**

**if(**ctx**.**isPointInPath**(**mouseFinX**,** mouseFinY**))**

ctx**.**fillStyle **=** "black"**;**

**else**

ctx**.**fillStyle **=** "white"**;**

ctx**.**font **=** "30px Georgia"**;**

ctx**.**fillText**(**"Joacă din nou!"**,** canvas**.**width **\*** 0.5**,** canvas**.**height **\*** 0.5 **+** 165**);**

**}**



În caz contrar, se desenează scorul și timpul, iar dacă este apăsat click și o săgeată nu este deja în aer, se desenează segmentul și statisticile asociate acestuia: puterea și unghiul.

**else** **{**

drawScoreAndTimeLeft**();**

gameTime -= 1/framesPerSecond;

**if(**mousePressed **&&** **!**arrowInAir**)** **{**

alfa **=** Math**.**atan2**(**mouseFinY**-**mouseInitY**,** mouseInitX**-**mouseFinX**);**

v0 **=** Math**.**min**(**euclidianDistance

**(**mouseInitX**,**mouseInitY**,**mouseFinX**,**mouseFinY**),** distMax**)** **/** **(**distMax**/**vmax**);**

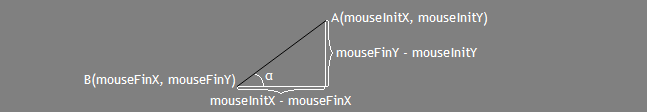
power **=** **(**v0**/**vmax**\***100**).**toFixed**(**2**);**

drawMouseSegment**();**

drawStats**();**

**}**

**}**

Pentru calcularea unghiului alfa se aplică funcția trigonometrică inversă arctangentă pe raportul dintre diferența pe verticală și pe orizontală între poziția finală și cea inițială a mouse-ului.

Deoarece originea este colțul din stânga-sus, coordonata y **crește** atunci când coboară pe ecran, de aceea diferența pe y trebuie sa fie invers față de cea pe x.

Pentru calcularea vitezei inițiale s-a folosit funcția Math**.**min pentru a-i pune o limită maximă (distMax) distanței dintre 2 puncte. Aceasta se împarte la raportul dintre *distanța maximă* si *viteza maximă* pentru a asigura atingerea vitezei maxime atunci când segmentul are distanța maximă.

Puterea este exprimată în procente, și reprezintă raportul dintre *viteza inițială* si *viteza maximă*.

Funcția drawMouseSegment desenează un segment care începe la poziția inițială a mouse-ului. Dacă lungimea segmentului este mai mică decât distMax segmentul se termină la poziția finală a mouse-ului. Dacă este mai mare, atunci separăm distanța maximă pe x și pe y, adică pe cosinus și sinus de alfa.

**function** drawMouseSegment**()** **{**

ctx**.**beginPath**();**

ctx**.**moveTo**(**mouseInitX**,** mouseInitY**);**

**if(**euclidianDistance**(**mouseInitX**,**mouseInitY**,**mouseFinX**,**mouseFinY**)** **>** distMax**)** **{**

mouseFinX **=** mouseInitX **-** distMax **\*** Math**.**cos**(**alfa**);**

mouseFinY **=** mouseInitY **+** distMax **\*** Math**.**sin**(**alfa**);**

**}**

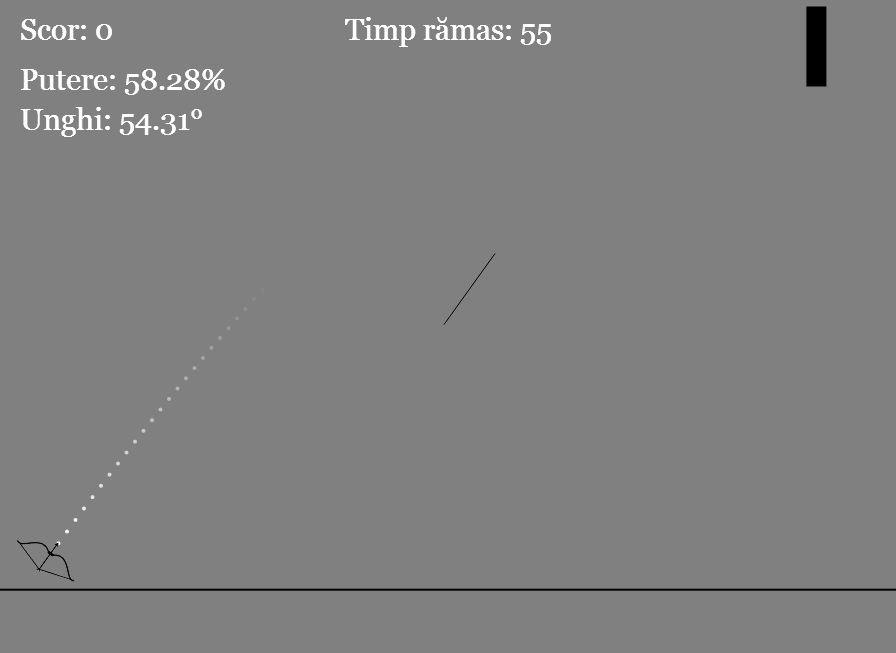
ctx**.**lineTo**(**mouseFinX**,** mouseFinY**);**

ctx**.**strokeStyle **=** 'black'**;**

ctx**.**stroke**();**

**}**

Desenarea segmentului, a puterii și a unghiului în funcție de proprietățile segmentului:



Se desenează ținta și săgețile înfipte, și se modifică poziția țintei în următorul cadru în funcție de viteza acesteia, totodată detectându-se coliziunea cu marginea de sus și podeaua:

**function** drawTargetAndArrows**()** **{**

ctx**.**fillStyle **=** "black"**;**

ctx**.**fillRect**(**targetX**,** targetY**,** targetWidth**,** targetHeight**);**

**for(var** nr **=** 0**;** nr **<** arrowsStuckTarget**.**length**;** nr**++)**

drawImage**(**arrow**,** arrowsStuckTarget**[**nr**].**x**,** arrowsStuckTarget**[**nr**].**y **+** targetY**,** 0.5**,** arrowsStuckTarget**[**nr**].**angle**);**

**for(var** nr **=** 0**;** nr **<** arrowsStuckFloor**.**length**;** nr**++)**

drawImage**(**arrow**,** arrowsStuckFloor**[**nr**].**x**,** arrowsStuckFloor**[**nr**].**y**,** 0.5**,** arrowsStuckFloor**[**nr**].**angle**);**

targetY **+=** targetSpeed**;**

**if(**targetY **<** 0 **||** targetY **+** targetHeight **>** podeaY**)**

targetSpeed **=** **-**targetSpeed**;**

**}**

Se parcurge cu câte o structură repetitivă de tip for fiecare vector și se desenează imaginea săgeata. Din păcate nu există nicio procedură ce rotește o imagine fără a afecta întregul canvas. Dar putem folosi ctx.setTransform pentru a muta punctul de origine și a putea redimensiona o imagine (în cazul acestui joc toate imaginile sunt redimensionate la 50% din valorile inițiale). Punctul de origine este mutat în centrul imaginii, și putem roti întregul fundal folosind ctx.rotate. Imaginea este desenată cu ctx.drawImage, iar apoi contextul revine la starea inițială setând scara de mărime la 1 și originea înapoi în punctul (0,0). Funcția corespunzătoare este:

**function** drawImage**(image,** x**,** y**,** scale**,** rotation**)** **{**

ctx**.**setTransform**(**scale**,** 0**,** 0**,** scale**,** x**,** y**);**

ctx**.**rotate**(**rotation**);**

ctx**.**drawImage**(image,** **-image.**width **/** 2**,** **-image.**height **/** 2**);**

ctx**.**setTransform**(**1**,**0**,**0**,**1**,**0**,**0**);**

**}**

Dacă nu este posibilă pregătirea unei noi săgeți (nu este apăsat pe mouse sau o săgeată este deja în aer sau jocul s-a sfârșit) este desenată imaginea cu arcul neîncărcat. Dacă o săgeată este în aer, atunci aceasta este desenată:

**if(!**mousePressed **||** arrowInAir **||** gameOver**)** **{**

drawImage**(**sprite**[**0**],** pozArcX**,** pozArcY**,** 0.5**,** **-**alfa**);**

**if(**arrowInAir**)**

drawArrow**();**

**}**

Desenarea unei săgeți este realizată prin apelul subprogramului drawArrow.

Pentru desenarea corectă a săgeții în aer, trebuie să știm unghiul vectorului de viteză la fiecare moment desenat. Știind că viteza pe orizontală nu se modifică (este constantă), obținem egalitatea:

Unde v reprezintă viteza la momentul t iar θ unghiul vectorului de viteză la momentul t. Astfel, putem afla unghiul θ din egalitatea:

Viteza la momentul t este determinată folosind formula:

Atunci când viteza pe verticală devine negativă, adică săgeata coboară, unghiul θ ar trebui să aibă o valoare negativă (pentru că vectorul vitezei ajunge sub axa orizontală). Pentru că nu putem folosi vectori, nu putem ști când trebuie să schimbăm semnul unghiului, dar știm că săgeata coboară când trece de **jumătatea distanței orizontale,** dacă punctul de aruncare și cel de atingere de sol sunt la același nivel. Pentru a generaliza rezultatul, **săgeata este în coborâre atunci când diferența dintre 2 coordonate y consecutive este pozitivă**, (de unde rezultă prevY > currentY):

**function** drawArrow**()** **{**

**var** v **=** Math**.**sqrt**(**v0 **\*** v0 **+** g **\*** g **\*** currentTime **\*** currentTime **-** 2 **\*** v0 **\*** g **\*** currentTime **\*** Math**.**sin**(**alfa**));**

**var** theta **=** Math**.**acos**(**v0 **/** v **\*** Math**.**cos**(**alfa**));**

currentY **=** coord\_y**(**currentTime**);**

**if(!**passedMiddle **&&** prevY **>** currentY**)**

passedMiddle **=** **true;**

Coordonatele a 2 momente consecutive sunt salvate:

prevY **=** currentY**;**

currentX **=** nextX**;**

currentTime **+=** 10 **/** framesPerSecond**;**

nextX **=** coord\_x**(**currentTime**);**

Detectarea coliziunii săgeții cu podeaua. La impact, coordonatele și unghiul final de rotație ale săgeții sunt adăugate în vector-ul arrowsStuckFloor[]:

**if(**pozArcY **-** currentY **>** podeaY**) {**

arrowsStuckFloor**.**push**({**x**:** pozArcX **+** currentX**,**

y**:** pozArcY **-** currentY**,** angle**:** theta**});**

arrowInAir **=** **false;**

**}**

Dacă între coordonatele pe orizontală ale celor 2 momente consecutive se găsește „x”-ul țintei atunci se apelează funcția collisionDetection:

**if(**currentX **+** pozArcX **<=** targetX **&&** nextX **+** pozArcX **>=** targetX**)**

**if(**collisionDetection**(**currentTime **-** 10 **/** framesPerSecond**))**

arrowInAir **=** **false;**

Funcția collisionDetection testează fiecare pixel pentru a asigura lovirea săgeții în țintă. Are un parametru time de unde începe structura repetitivă de tip for, oprindu-se când ajunge la timpul prezent. La coliziune:

* se adaugă coordonatele și unghiul final de rotație beta (în funcție de variabila passedMiddle aceasta va fi negativă sau pozitivă) - utilizând aceeași formulă ca în cazul unghiului theta - ale săgeții în vector-ul arrowsStuckTarget[],
* se acordă puncte (tot în funcție de variabila passedMiddle),
* se returnează valoarea true:

**function** collisionDetection**(**time**)** **{**

**for(var** Y **=** time**;** Y **<=** currentTime **;** Y **+=** 0.0001**)**

**if(**pozArcY **-** coord\_y**(**Y**)** **>** targetY **&&** pozArcY **-** coord\_y**(**Y**)** **<** targetY **+** targetHeight **&&** pozArcX **+** coord\_x**(**Y**)** **>=** targetX**)** **{**

**var** v2 **=** Math**.**sqrt**(**v0 **\*** v0 **+** g **\*** g **\*** Y **\*** Y **–**

2 **\*** v0 **\*** g **\*** Y **\*** Math**.**sin**(**alfa**));**

**var** beta **=** Math**.**acos**(**v0 **/** v2 **\*** Math**.**cos**(**alfa**));**

**if(!**passedMiddle**){**

score**++;**

arrowsStuckTarget**.**push**({**x**:** pozArcX **+** coord\_x**(**Y**),**

y**:** pozArcY **-** coord\_y**(**Y**)** **-** targetY**,** angle**:** **-**beta**});**

**}**

**else** **{**

score **+=** 2**;**

arrowsStuckTarget**.**push**({**x**:** pozArcX **+** coord\_x**(**Y**),**

y**:** pozArcY **-** coord\_y**(**Y**)** **-** targetY**,** angle**:** beta**});**

**}**

**return** **true;**

**}**

**return** **false;**

**}**

Se verifică dacă săgeata a trecut de marginile orizontale:

**if(**pozArcX **+** currentX **<** 0**)**

arrowInAir **=** **false;**

**if(**pozArcX **+** currentX **>** canvas**.**width**)**

arrowInAir **=** **false;**

Dacă săgeata încă mai este în aer după verificarea tuturor condițiilor de mai sus, atunci săgeata poate fi desenată în aer prin subprogramul drawImage, unghiul de rotație theta fiind negativ dacă săgeata este în urcare sau pozitiv dacă săgeata este în coborâre:

**if(**arrowInAir**)** **{**

**if(!**passedMiddle**)**

drawImage**(**arrow**,** currentX **+** pozArcX**,** pozArcY **-** currentY**,** 0.5**,** **-**theta**);**

**else**

drawImage**(**arrow**,** currentX **+** pozArcX**,** pozArcY **-** currentY**,** 0.5**,** theta**);**

**}**

**}**

Revenind la subprogramul draw, dacă este posibilă pregătirea unei noi săgeți, atunci este desenată o traiectorie ajutătoare ce devine din ce în ce mai transparentă (pentru a nu face jocul prea ușor) folosind o structură repetitivă de tip for și subprogramul drawPoint ce desenează un „punct” (un cerc cu raza de 2 pixeli) la coordonatele specificate. Această traiectorie se sfârșește fie când traiectoria „atinge” podeaua, fie când opacitatea traiectoriei devine 0 (transparență completă).

În funcție de puterea arcului, se desenează imaginea corespunzătoare. Avem 10 imagini („tile001” – „tile011”) cu arcul și săgeata în diferite stadii, deci vom realiza distribuția imaginilor în intervale de 10. Împărțind puterea la numărul de imagini obținem 100/10 = 10. Pentru a nu scrie un „switch” cu 10 cazuri diferite, am realizat această împărțire în indicele vectorului, luându-i doar partea întreagă și adunând-o cu 1. Imaginea este desenată la coordonatele arcului pozArcX și pozArcY, și este rotită la opusul unghiului alfa:

**else** **{**

**for(var** t **=** 0.1**;** **;** t **+=** 0.1**)** **{**

toDrawX **=** coord\_x**(**t**);**

toDrawY **=** coord\_y**(**t**);**

**if(**pozArcY **-** toDrawY **>** podeaY **||** 1 **-** **(**toDrawX**+**toDrawY**)\*** 0.002 **<=** 0**)**

**break;**

drawPoint**(**toDrawX **+** pozArcX**,** pozArcY**-**toDrawY**,**1**-(**toDrawX **+** toDrawY**) \*** 0.002**);**

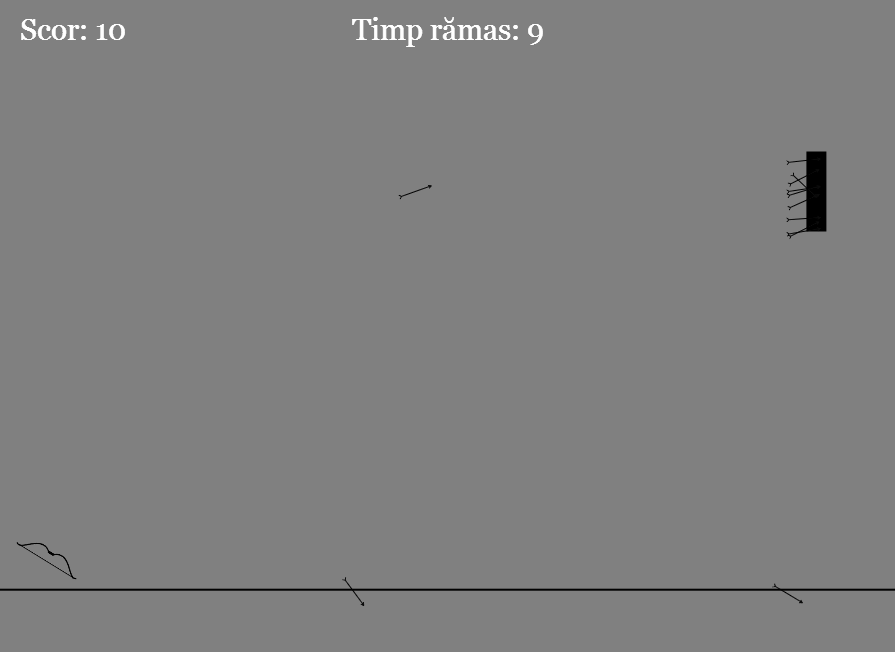
**}**

drawImage**(**sprite**[**Math**.**floor**(**power **/** 10**)** **+** 1**],**pozArcX**,**pozArcY**,** 0.5**,** **-**alfa**);**

**}**

**}**

Un exemplu de cadru cu săgeți înfipte și o săgeată în aer:



III.Bibliografie

<http://ro.nccmn.wikia.com/wiki/Formule_de_mecanică>, Nicolae Coman

<https://www.w3schools.com/tags/>, W3Schools

<https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D>, MDN Web Docs