SIMULACIÓ D'UN TIR PARABÓLIC AMB UN APPLET DE FÍSICA

Alumne: Josep Fàbregas Cuadrada

ETIG / ETIS

Consultor: Viçens Font Sarista

Data de lliurament: 14/6/16

Aquest projecte té una vesant clarament pedagògica, en un intent d'apropar l'ensenyament a l'aula a partir de la informàtica en les seves infinites aplicacions.
En particular l'assignatura que es pretén portar a l'aula des de la informàtica es la Física. I de forma més concreta un tema anomenat cinemàtica i amb més precisió el 'tir parabòlic'.

L'objectiu principal ha estat crear un applet de simulació d'un tir parabòlic, per tal de poder ser observat i descrit a classe. I per facilitar als alumnes l'estudi des de casa amb l'observació e interacció amb el mateix.

Per tant s'ha dissenyat i programat un aplicatiu amb JAVA en un entorn visual. L'especialitat informàtica a la que se m'ha reconduït ha estat J2EE.

El programa no té una gran complexitat en quan al model interactiu en xarxa. La dificultat ha estat programar la matemàtica que conté la física estudiada, el disseny visual dels objectes en moviment i la implementació de la interacció usuari-software en el moviment i control del mateix.

Josep Fàbregas

INDEX

1. Introducció

L'aplicació que vull desenvolupar es troba definit en l'àmbit de la física i la matemàtica. Per tant es vol implementar un programa que contingui un alt nivell de càlcul matemàtic i d conceptes vinculats amb la física.
El camp en que es vol dirigir l'aplicació és pedagògic més que científic. És a dir, un software que tingui como a funció l'aprenentatge de física.

En particular he escollir una part de la física que es dóna a 1r i 2n de batxillerat i a 1r de carreres tecnològiques com física, enginyeria industrial, telecomunicacions, camins canals y ports, etc. El software també pot ser útil a aquest nivell universitari.
Vull fer un software de cinemàtica que representi un 'tir parabòlic' o 'llançament d'un projectil'. És a dir, un tir lliure bidimensional amb gravetat constant, que consta de dos moviments en els eixos 'x' i 'y'. Un moviment rectilini uniforme (MRU) en 'x' i un moviment rectilini uniformement accelerat (MRUA) en l'eix 'y'.

La idea principal seria emular o simular un 'tir parabòlic', en un applet en moviment en funció del temps, de forma que el projectil es pogués visualitzar en una simulació.

2. El tir parabòlic en context

Hem utilitzat el llenguatge JAVA. Es desenvoluparà l'aplicatiu en la plataforma Eclipse-Luna. L'aplicatiu suporta diversos entorns doncs JAVA es pot utilitzar en diferents sistemes operatius com Windows, Linux, Unix, Ferdora, Ubuntu, Debian, Kali...

Te però certes restriccions com Android o IOS. Tot i que avui en dia ja existeixen aplicacions que simulen sistemes operatius en IOS, amb la qual cosa també seria implementable.
La raó és que JAVA no suporta aquests sistemes operatius mentre que amb JAVA-scrip si hauria estat possible. La idea és que aquest aplicaria sigui útil en ordinadors més que en mòbils o IPAD's ja que té una orientació clarament pedagògica.

El llenguatge JAVA, és dels més utilitzats, assegurant fiabilitat i robustesa. Sent el més utilitzat per fer Applets i en particular Phislets (Applets aplicats a la física)
La idea és incrustar el JAVA en HTML per tal que sigui accessible a una pagina WEB.

L'aplicatiu és molt simple en la seva definició, té un arxiu principal amb dues classes, per tant 2 nivells de programació. La dificultat rau en el gran nombre de variables físiques que entren en joc i en la matemàtica implicada per fer els càlculs.

3. Objectius

Crec que seria bo anomenar-los per donar una diversitat sobre la fita del treball.

a. Crear un software de qualitat

Es pretén crear un software de qualitat, robust, funcional i eficient

b. Crear un software pedagògic

Es pretén crear un programari útil per l'ensenyança, que pugi ser tant explicat a classe como ser visualitzat des de casa per poder estudiar.

c. Crea un software a partir d'una metodologia universitària

S'ha intentat crear un software a partir d'assignatures estudiades a classe como Orientació a Objectes o Enginyeria del programari. És a dir s'ha intentat produir-lo de forma metodològica en base als estudis previs que s'han cursat en la carrera. Tant en el disseny, com en la creació com en l'excussió.

d. Crear un software versàtil

També s'ha intentat crear un software que sigui ampliable. La creació orientada a objectes permetria amb relativa facilitat crear altes classes que ampliessin el domini i l'abast del projecte introduint millores substancials al producte inicials. Això como poder implementar altres mètodes per millorar-ne la qualitat pedagògica.

e. Crear una base de programació per altres APPLETS

El software creat podria ser una base per fer altes APPLETS. Un codi base per construir altres simulacions. O como a mínim com experiència per crear-ne d'altres més eficients.

f. Crear un software comercial

Sense ser un objectiu principal ni prioritari. Tampoc negaré que he pensat de crear un conjunt d'APPLETS de física orientat a la física de 1r i 2n batxillerat i oferir-los a la venta.
Per això òbviament s'hauria de millorar de forma considerable el producte. I implementar-los també en altres llenguatges com Android o IO's, per tal de ser visualitzats en mòbils i IPAD's.

4. Enfocament i mètode seguit

- **Enfocament:**
  S'ha seguit una metodologia de creació inspirada en l'orientació a objecte l'estudi casos d'us de l'assignatura Enginyeria del programari.

- **Entorn de programació**
  L'entorn de programació ha de ser visual. Un entorn gràfic que permeti desenvolupar un simulació en moviment i doni una sensació de realisme al usuari.
  JAVA és un llenguatge adaptat a un entorn visual (visual JAVA). Molt escaient per aquest tipus de programari, per això la majoria d'applets estan fets en JAVA, tot i que s'estan desenvolupant d'altres llenguatges (arduino, pyton, perl, android...) implementats en sistemes operatius de mòbils i ipad's que comencen a ser una competència molt real.

- **Requisits de caràcter pedagògic.**
  Es busca que l'aplicació sigui 'amigable', de fàcil ús, intuïtiva i que no generi rebuig se visualitzada. Amb un entorn 'atractiu' que generi un 'desig' de ser utilitzat.

  Similar a les aplicacions de jocs, però òbviament amb una finalitat pedagògica i no lúdica.
  L'aplicació ha de ser eminentment pedagògica, doncs la finalitat és 'aprendre física' de forma no clàssica, sinó interactiva. No es busca un tipus d'aprenentatge 'tipus classe magistral', sinó un aprenentatge interactiu que es generi per la utilització de l'aplicació.
  Per això es requereix un nivell de interacció fàcil i amè amb l'usuari (alumne o estudiant)

- **Requisits de caràcter tècnic.**
  Com ja hem dit ha de ser un entorn visual. El programa ha de ser robust, fiable, segur, eficient en la utilització de recursos (tan de memòria RAM, com del processador). No ha de tenir redundància de codi i s'escollirà entre diferents mètodes gràfics que conté JAVA.

  M'inclino per la classe Thread t = new Thread() com a variable temporal i el mètode graph de JAVA paint(Graphics g), amb el mètode repaint().

  L'aplicació s'haurà de poder executar remotament des de qualsevol màquina (alumnes estudiants) i per tant haurà de ser viable de ser incrustat en un llenguatge d'internet HTML, per exemple.

5. Planificació del projecte

La planificació ha intentat ser ordenada, metòdica i sistemàtica. Això s'han anat assolint fites seguint un guió predeterminat i prèviament dissenyat.

Aquestes han estat les fites que s'han anat assolint en una planificació sistematitzada

5.1 Concreció de l'ús de llenguatge i tipus mètodes gràfics.

5.2 Concepció matemàtica i física. Disseny matemàtic teòric.

5.3 Concepció i disseny de classes i objectes.

5.4 Disseny de l'esquelet, botons i eixos de coordenades.

5.5 Disseny i implementació de l'entrada de dades.

5.6 Implementació del moviment bàsic de la partícula (oval).

5.7 Disseny dels vectors. Classe.

5.8 Implementació d'objectes visuals en moviment: vectors de posició, vectors de velocitat i vectors d'acceleració.

5.9 Implementació del control del moviment

5.10 Control d'errors i joc de proves

5.11 Implementació en HTML.

5.12 Implementació en un servidor.

Temporització de lliuraments:

La planificació a nivell de timing prevista ha estat aquesta. Tot i que els lliuraments no s'han produït en els temps indicats. La producció ha estat seguida amb bastanta fidelitat en base al que s'havia planificat.

Temporització- Calendari

		DLL	DM	DM	DJ	DV	DS	DG	FITES	HORES	LLIURAMENTS
MARÇ	SET-1		1	2	3	4	5	6	1. Llenguatge, mètode gràfic		
	SET-2	7	8	9	10	11	12	13	2. Concepció matemática i física disseny teòric	8h	
	SET-3	14	15	16	17	18	19	20	3. Concepció i disseny de classes i objectes	5h	
	SET-4	21	22	23	24	25	26	27	4. Disseny de l'esquelet, botons i eixos	5h	botons i estructura básica
	SET-5	28	29	30	31	1	2	3	5. Disseny i implementació de l'entrada de dades	8h	
ABRIL	SET-6	4	5	6	7	8	9	10	6. Implementació del moviment básic de la partícula	6h	moviment partícula
	SET-7	11	12	13	14	15	16	17	7. Disseny de la classe vectors	10h	
	SET-8	18	19	20	21	22	23	24	8. Implementació d'objectes visuals, vectors	10h	
	SET-9	25	26	27	28	29	30	1	8. Implementació d'objectes visuals, vectors	10h	moviment vectors
MAIG	SET-10	2	3	4	5	6	7	8	9. Implementació del control del moviment	10h	control del moviment
	SET-11	9	10	11	12	13	14	15	10. Control d'errors / joc de proves	6h	
	SET-12	16	17	18	19	20	21	22	11. Implementació HTML / 12. Implementació servidor	8h	Applet acabat
	SET-13	23	24	25	26	27	28	29			
JUNY	SET-14	30	31	1	2	3	4	5			
	SET-15	6	7	8	9	10	11	12			
	SET-16	13	14								

6. Producte obtingut

El producte obtingut és aparentment molt simple, un aplicaria d'unes mil línies de codi. La intenció es poder-lo exposar en una web incrustat en un html, per ser accessible remotament i que faci el degut servei a les persones que s'hi volen connectar.

7. Estructura de l'aplicatiu

7.1. Estructura visual

Explicarem l'aplicatiu descrivint tots els botons i finestres de que consta el programa. Inicialment capturem una imatge de l'escenari que observem a l'obrir l'aplicatiu.
Les imatges que es veuran seran:

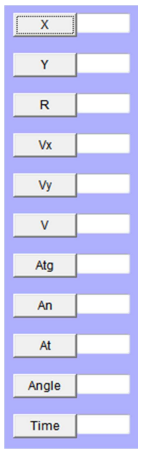
- Els botons d'informació i finestres amb les dades en moviment.
- Els botons i finestres d'entrada de dades.
- Els botons de control del moviment
- La partícula en moviment (oval), el rastre i els eixos de coordenades.
- Els vectors posició, velocitat i acceleració.

Aquesta seria una primera imatge sense cap tipus de moviment.

Passarem a descriure cadascun del objectes visuals:

- <u>Botons informatius i finestres de dades físiques rellevants</u>

  Tenim 11 botons informatius (no actius) a la dreta del disseny amb 11 finestres on es mostra la informació del moviment del tir parabòlic. A saber:

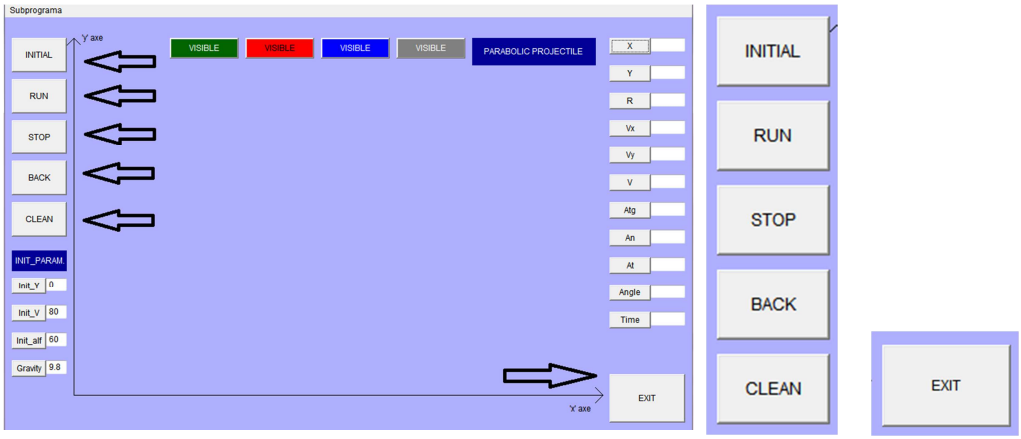
  

  - Posició x: X
  - Posició y: Y
  - Radi vector: R
  - Velocitat en x: Vx
  - Velocitat en y: Vy
  - Mòdul total de la velocitat: V
  - Acceleració tangencial: Atg
  - Acceleració normal: An
  - Acceleració total (acceleració de la gravetat) : At
  - Angle: Angle
  - Temps transcorregut: Time

  Si l'usuari escriu en les finestres. El sistema no nota aquests canvis. Els valors que apareixeran posteriorment seran els valors per defecte que el software calcula.

- <u>Botons de control del moviment</u>

  Tenim 6 botons de control. Potser el més important.
  INITIAL, RUN, STOP, BACK, CLEAN, EXIT.

  El 'valor afegit' que ofereixen els applets, és justament el poder interaccionar amb la física en una simulació. Redirigint el procés en la direcció que l'usuari desitja. D'aquí que aquest fet sigui un factor clau per l'aprenentatge. Això permet a l'alumnat comprendre millor la física que se'n desenvolupa.

Així doncs:

- **INITIAL**: El sistema no funciona fins que no es prem el botó INITIAL i es tanca amb el botó EXIT. Si s'interactua amb qualsevol dels restants botons (RUN, STOP, BACK, CLEAN), els sistema no s'altera.
  INITIAL et col·loca el mòbil en el punt inicial en repòs (origen de coordenades), on aquest punt pot ésser alterat per l'entrada de dades efectuada per part de l'usuari. En qualsevol moment en que premem INITIAL els sistema tornarà a les condicions inicials que hi ha per defecte o les entrades per l'usuari. Tornant a la situació inicial de repòs.
- **EXIT**: L'aplicatiu s'apaga.
- **RUN**: El moviment comença i totes les finestres de la dreta s'omplen de valors en moviment, resultat dels diferents càlculs 'en vol' fets pel programa. La interacció de l'usuari amb les finestres, tant de la dreta, com de les dades inicials no altera el moviment. En els de la dreta, es re-adrecen immediatament i en els de l'esquerra, queden gravats i seran modificats en el proper moviment en que es premi INITIAL.
- **STOP**: El sistema permet aturar 'en vol' el moviment, observant els diferents paràmetres a estudiar ara estàtics. Així com els vectors posició, velocitat i acceleració.
- **BACK**: Els sistema permet una reversibilitat en el temps, és a dir, fer anar el moviment cap endavant naturalment o fer-lo retrocedir, si es vol tornar a una posició desitjada. Així doncs els paràmetres de la dreta van en sintonia amb el procés cap endavant o cap endarrere. Es notable destacar que el temps va cap endarrere en aquest mode.
  Si el sistema es deixa anar totalment endarrere, es situa en la posició inicial.
- **CLEAN**: Esborra la pantalla, el moviment que hi ha. En qualsevol moment es pot esborrar. Si es prem RUN continua en la mateix posició. Si es prem INITIAL es comença un altre cop el moviment.

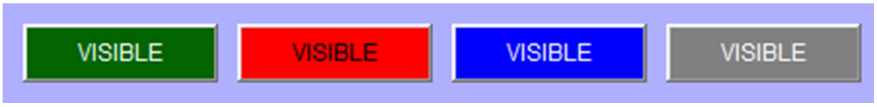
- L'aplicaitu s'atura en l'abast màxim del tir (la posició horitzontal màxima), doncs té un valor físic rellevant que es pot observar en la casella X. Si s'activa RUN continua. I si s'activa BACK torna enrere.
- L'aplicatiu pot anar endavant i enrere indistintament i actua immediatament segons l'ordre donada.
- Si es premen diferents botons como RUN, BACH, STOP, RUN... els sistema obeeix a l'usuari.

➢ <u>Efectes visuals</u>

Els efectes visuals no son un simple 'efecte atractiu' en el moviment. Son molt rellevants a nivell físic.

Doncs el model cinemàtic, es basa en l'existència de vectors com a tal. El model matemàtic del tir parabòlic és per tant vectorial. Tot i que (òbviament) els vectors no es veuen en la realitat, la física en suposa la seva existència. Basant el seu 'model predictiu' en una descripció dels mateixos.

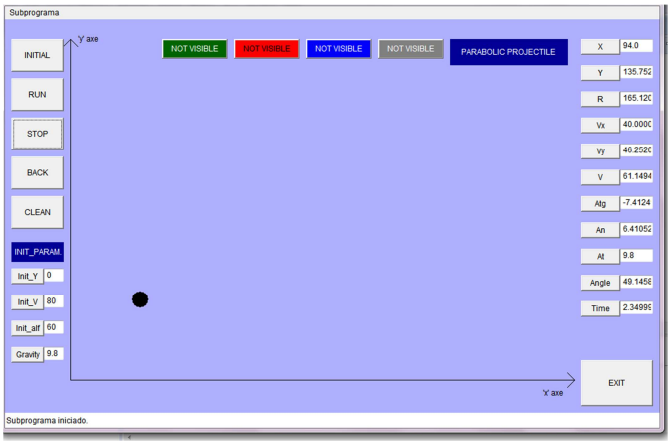
Aquests serien els botons per controlar els efectes visuals.



- Esfera o punt

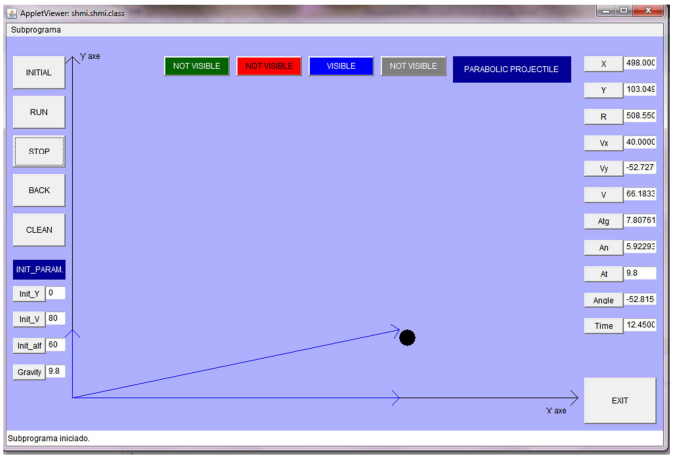
  Tot l'aplicatiu es basa en un punt (esfera) que simula el moviment, però ve acompanyat de diferents efectes que descriuen el moviment.

  El punt no es optatiu, es totalment necessari, sino el propi moviment no tindria sentit.



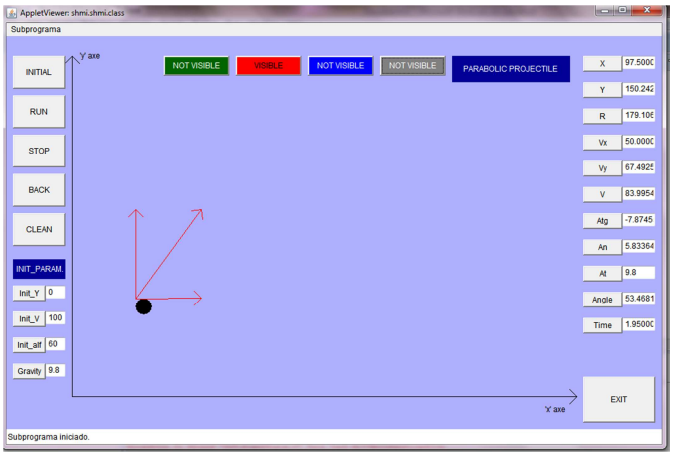
- Vector posició en blau

  El vector posició en blau ens descriu el moviment respecte de l'origen de referència. És optatiu de forma que l'usuari pot fer que apareix o no. Per defecte apareix.



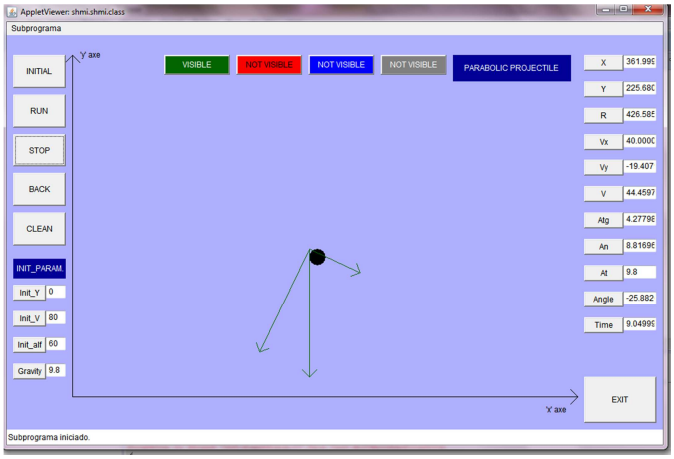
- Vector velocitat en vermell

  El vector velocitat dóna la velocitat en x i en y i es optatiu. Apareix per defecte.
  I pot variar la seva longitud segons siguin els valor inicials entrats.
  És important notar que aquesta longitud és directament proporcional als valor que apareixen en les finestres Vx, Vy, V.



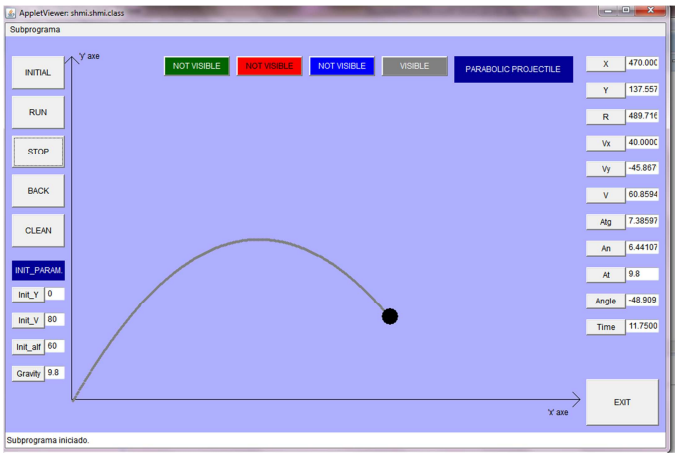
- Vector acceleració en verd

  Dóna el valor de l'acceleració, tant l'acceleració normal com l'acceleració tangencial i la suma de les dues, que serà la gravetat, un valor constant. És optatiu i apareix per defecte. És important notar que el valor de les tres acceleracions apareix en les finestres Atg, An, At. La seva longitud (adaptable segons el software definit) és directament proporcional als valors que apareixen a la dreta. És important notar que el vector vertical (la gravetat) és sempre constant en el moviment.



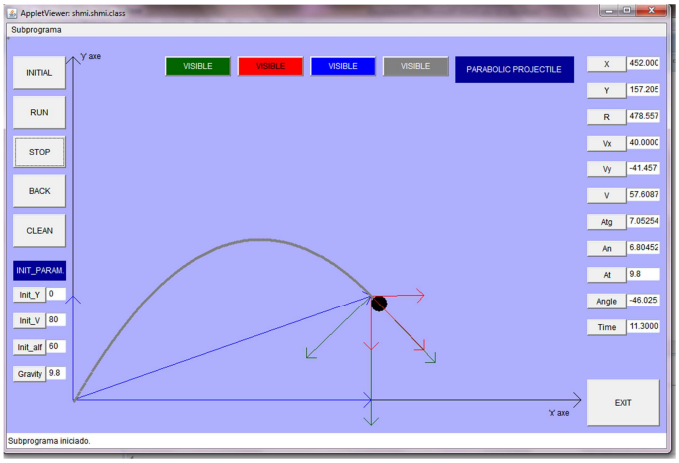
- Trajectòria en gris

  Indica els punts pels quals ha passat el mòbil, deixant un rastre imaginari en l'espai. Es optatiu i apareix per defecte.



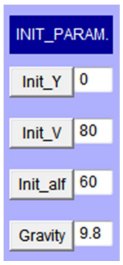
El fet que els efectes visuals siguin opcionals, permet que l'usuari o alumne tingui una perspectiva específica de cada variable o paràmetre a estudiar, ajudant a l'aprenentatge i comprensió dels conceptes explicats.

Aquí tenim una captura de tots els efectes visuals activats simultàniament.



➢ <u>Finestres d'entrada de dades</u>

Tenim 4 botons d'entrada de dades. Amb les 4 respectives finestres. Això permetria a l'usuari crear les condicions inicials (en física condicions de contorn) que determinarien el tir. Per defecte tenim Init_Y = 0, Init_V = 80, Init_alf = 60, Gravity = 9.8.



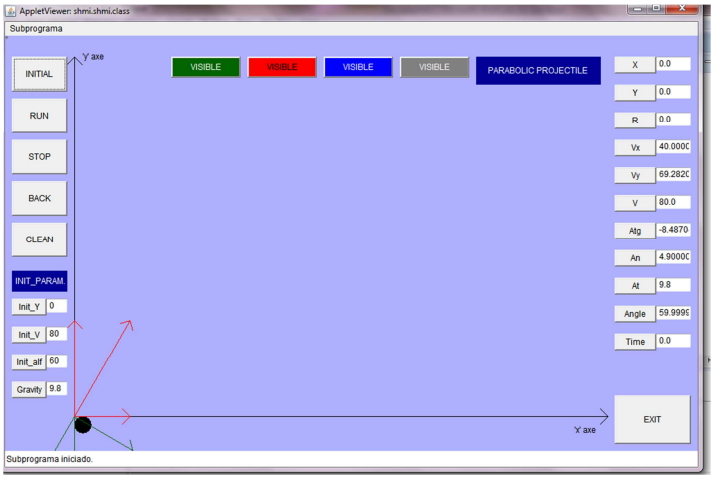
- Altura inicial: Init_Y
- Velocitat inicial en mòdul: Init_V
- Angle inicial: Init_alf
- Gravetat del "planeta" escollit: Gravity

Si l'usuari no interactua amb aquestes finestres, el tir es desenvoluparia normalment amb el valors per defecte.

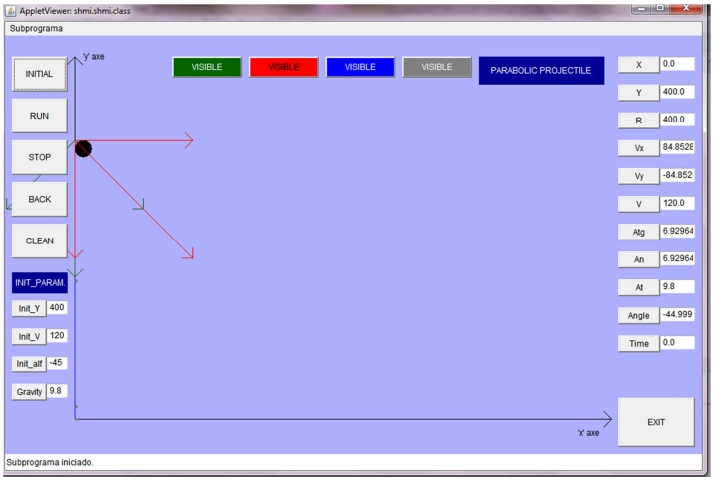
El marc INIT_PARAM en blau marí és explicatiu. Informant a l'usuari sobre a on s'ha d'entrar les dades inicials.

Aquí es mostren diferents altures, angles i velocitats inicials, variant les dades inicials per part de l'usuari. És interessant notar que la gravetat també és adaptable.

- Posició inicial per defecte:



- Posició introduïda per l'usuari, altura inicial, velocitat inicial i angle inicial

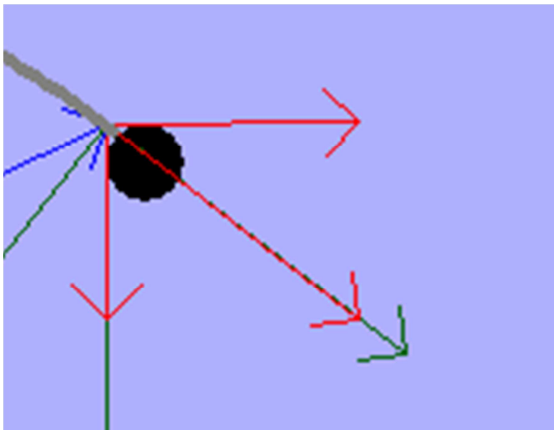


➢ <u>Disseny dels vectors</u>

Mereix, sense dubte, una menció a part el disseny de vectors (les fletxetes).

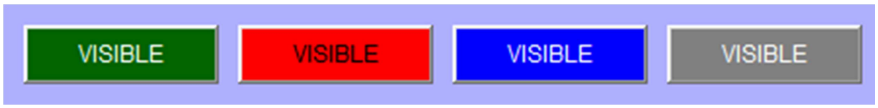
Ha estat una tasca bastant dificultosa, però crec que era fonamental donar un efecte visual adient en la descripció dels vectors. Les fletxetes de les puntes dels vectors, han estat dissenyades matemàticament i es referencien en una classe a part (classe vector), que segueix les puntes i els hi va donant una imatge de fletxa, formant 45º amb la recta base del vector, que dona la direcció.

Els problemes han estat diversos i m'ha ocupat un nombre d'hores realment considerable.



- Botons de control d'efectes visuals

  No ha estat especialment difícil implementar-ho. Tenim els següents booleans per controlar els efectes visuals.




```
int trajectory = 1;
int velocity = 1;
int acceleration = 1;
int position = 1;
```

- Control de les dades d'entrada

  S'han definit uns rangs, tant per l'altura inicial, velocitat inicial i acceleració. De forma que si l'usuari es surt del rang establert, el sistema emet un missatge demanant que es canviïn els valor i mostrant-ne el rang de validesa. El programa no es reactiva fins que els rangs entrats no siguin els correctes. I s'ha de tornar a clicar el botó del missatge.

Això afecta a les 4 entrades, amb els següents rangs o intervals permesos.
Altura: 0 < y < 500
Velocitat: 0 < v < 1000
Angle: -90 < alf < 90
Gravetat: 0 < gravity < 20.

7.2. Import Java

Dintre dels diferents mètodes de JAVA per gràfics (java.awt.Graphics) tenim a disposició, entre d'altres, aquestes llibreries, per l'entorn visual.

```
import java.lang.*;
import java.applet.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;
import java.awt.Graphics;
import java.io.*;
import java.awt.*;
import java.math.*;
import javax.swing.JButton;
import javax.swing.JPanel;
```

Dintre de totes aquestes possibilitats farem ús de:

```
import java.awt.Graphics;
import java.applet.*;
```

L'aplicatiu es farà amb la plataforma ECLIPSE-Luna.

7.3. Animació

El mètode utilitzat serà: "public void paint(Graphics g)"

Amb la variable temporal desenvolupada per JAVA: Thread t = new Thread().

Que serà la vertadera clau del la simulació de moviment.

7.4. Variables i classes

Hauríem d'especificar les variables de cada classe.

- Classe principal shmi de l'arxiu "shmi.java",

  Consta de les següents variables:

  - ✓ Una variables de l'entorn JAVA tipus Thread de control temporal.

    Thread t = new Thread()
  - ✓ Comptadors:

    int i

    int n
  - ✓ 5 variables d'inicialització del tir parabòlic. Determinades per defecte o entrades per l'usuari.

    double y0

    double v0

    double time

    double gravity

    double alf0

- ✓ 5 booleans de control, tipus “int” que funcionen en dos estats: 1,0.
  Serveixen per controlar l’estat de vol o a terra, en funcionament o aturat, borrat de les figures i estat enrere o endavant.
  int flying
  int running
  int stop
  int clean
  int back
- ✓ Un booleà per controlar els canvis de direcció dels vector acceleració.
  int dir
- ✓ 5 booleans tipus “int” amb estats 1,0, per controlar la visualització de la trajectòria, els vectors velocitat, els vectors acceleració i els vectors posició.
  int trajectory
  int velocity
  int acceleration
  int position
- ✓ Colors de pantalla
  Color greenColor = new Color()
  Color blueColor = new Color()
  Color darkBlueColor = new Color()
- ✓ Dues distàncies per determinar la longitud proporcional i relativa dels vectors velocitat i acceleració i de les fletxes.
  double k
  double k2
- ✓ Dos arrays per guardar la informació de les diferents posicions del mòbil en l’espai.
  int array_x[] = new int[10000]
  int array_y[] = new int[10000]
- ✓ La variable fonamental tipus particle, una classe que conté total la informació en moviment. Amb entrada, còmput i sortida de dades per processar el moviment de tot el tir parabòlic.
  particle p = new particle()
- ✓ Variables internes de moviment generades en el mètode paint(graphics g)
  - Variable “g”, de “paint(graphics g)” per obtenir els diferents colors, segments en moviment i dos ovals que seran la partícula en moviment i el rastre que deixa.
    - ✓ g.setColor()
    - ✓ g.drawLine()
    - ✓ g.fillOval()
  - Variables visuals tipus vector, amb accés a les fletxes en la classe vector.
    Tenim els vectors: posició x, posició y, radi r, velocitat en x, velocitat en y, velocitat total, acceleració tangencial, acceleració normal i acceleració total.
    vector x = new vector()
    vector y = new vector()

vector r = new vector()
vector vx = new vector()
vector vy = new vector()
vector v = new vector()
vector an = new vector()
vector at = new vector()
vector a = new vector()

- Classe particle de l'arxiu "particle.java".
  Tenim les següents variables, per calcular les crides des de l'arxiu principal.
  Tenim les variables posició inicial, velocitat inicial, gravetat, angle inicial i temps.
  double y0
  double v0
  double gravity
  double alf0
  double time

- Classe vector de l'arxiu "vector.java"
  - ✓ Té les variables d'entrada de posicions en x i en y dels 4 punts de les fletxes, per tant 8 punts:
    float x1,x2,y1,y2
    float y2r,y1r,x2r,x1r
  - ✓ Un booleà de direcció tipus int:
    int dir
  - ✓ Una variable de longitud de les fletxes:
    float L

7.5. Classes de JAVA i procediments

L'aplicatiu consta de 3 arxius .java:

a) shmi.java:

El principal anomenat "shmi.java", és la classe "public class shmi extends java.applet.Applet". Conté els mètode d'inici (main() en JAVA no visual) i de creació de figures geomètriques en moviment. Conté tots els botons i finestres, tant d'entrada i sortida de dades com de control per part de l'usuari.

Exactament té:

1. 15 botons estètics d'informació de paràmetres tipus "button".
2. 4 Finestres d'informació, tipus "label".
3. 10 botons interactius per manipular l'applet. Tipus "button".
4. 11 finestres d'informació de paràmetres físics. Tipus "textField".
5. 4 finestres d'introducció de dades per part de l'usuari. Tipus "textField".

Cada element visual en VISUAL_JAVA és un mètode. Per defecte tots ho són.

Tenim llavors els següents mètodes per executar l'aplicatiu:

a) public void init(){}:
   Equivalent a el void main() de JAVA. És a dir la subrutina principal a la que s'accedeix per defecte.
b) private void initComponents(){}:
   Inicialització de tots els botons i finestres de text.
c) public void paint(Graphics g){}:
   El mètode "paint" per crear figures en moviment en l'applet.
d) Tots els botons i finestres d'informació i interactives. Aquestes tenen un codi intern que reacciona amb el mètode paint(graphics g) al ser clickat, a partir de la crida repaint(). Això permet a l'usuari executar els canvis que vulgui fer 'en vol', interactuant amb l'applet.

b) particle.java:

Tenim un arxiu de càlcul matemàtic que conté tots els càlculs físics del moviment del projectil amb 27 mètodes per calcular totes les variables necessàries.

Tenim diferents mètodes per entrada de dades a la classe tipus set_ i d'obtenció de dades en crides des de l'arxiu principal tipus get_

Tenim 5 mètodes d'entrada de dades, on entrarem els paràmetres principals: Velocitat inicial, posició inicial, angle inicial, gravetat i temps i la resta, 22 mètodes, són càlculs de diferents paràmetres i variables físiques.

```
public void set_y0(double y0){}
public void set_v0(double v0){}
public void set_alf0(double alf0){}
public void set_gravity(double gravity){ }
public void set_time(double time){ }
```

public double get_time(){}
public double get_x(){}
public double get_y(){}
public double get_xw(){}
public double get_yw(){}
public double get_rw(){}
public double get_vx(){}
public double get_vy(){}
public double get_v(){}
public double get_alf(){}
public double get_alfw(){}
public double get_atg(){}
public double get_an(){}
public double get_at(){}
public double get_atgx(){}
public double get_atgy(){}
public double get_anx(){}
public double get_any(){}
public double get_atgw(){}
public double get_anw(){}
public double get_atw(){}
public double get_timeMax(){}

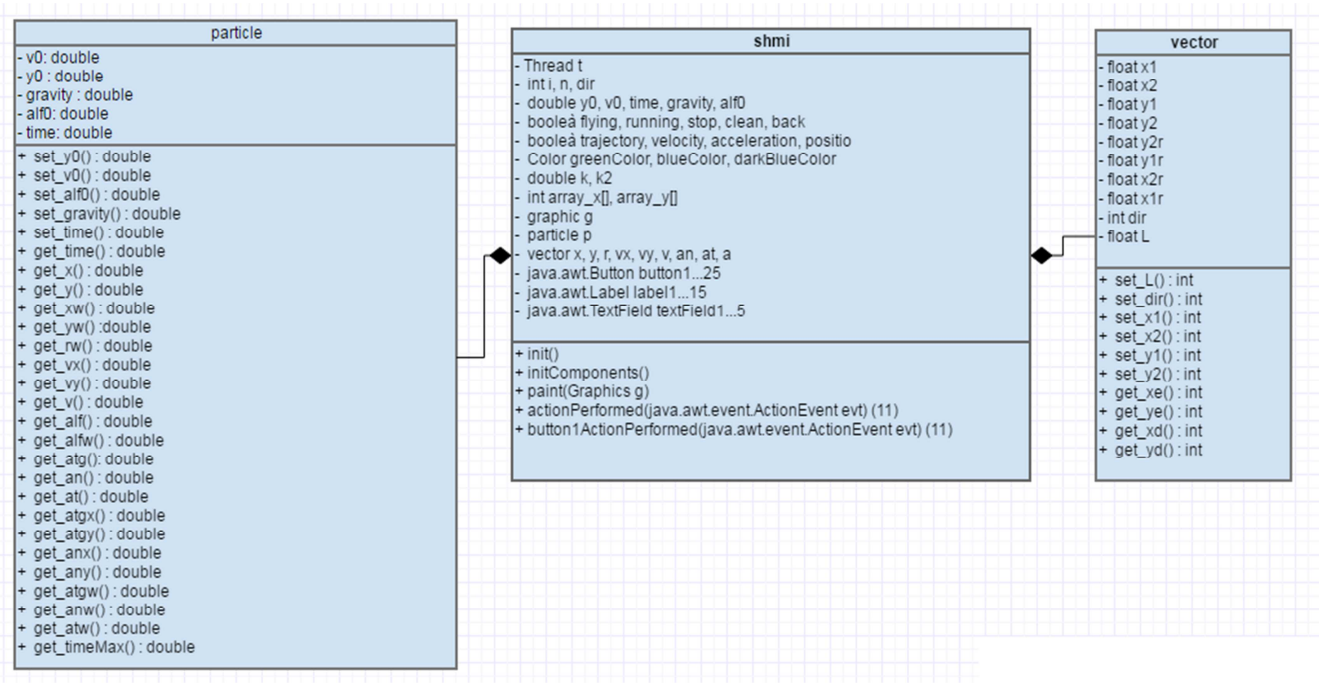
c) vector.java:

El tercer arxiu .java anomenat "vector.java" és un arxiu relegat específicament a càlculs purament geomètric i visuals dels diferents vectors que entren en joc. A saber: els de la posició, velocitat i acceleració. És a dir és un càlcul purament estètic de les fletxes, per donar la sensació verídica de ser vectors. Té una base matemàtica i geomètrica de càlcul.

Tenim 6 mètode d'entrada de paràmetres i 4 mètodes per calcular els 4 punts de les fletxes dels vectors.

public void set_L(int L){}
public void set_dir(int dir){}
public void set_x1(int x1){}
public void set_x2(int x2){}
public void set_y1(int y1){}
public void set_y2(int y2){}
public Integer get_xe(){}
public Integer get_ye(){}
public Integer get_xd(){}

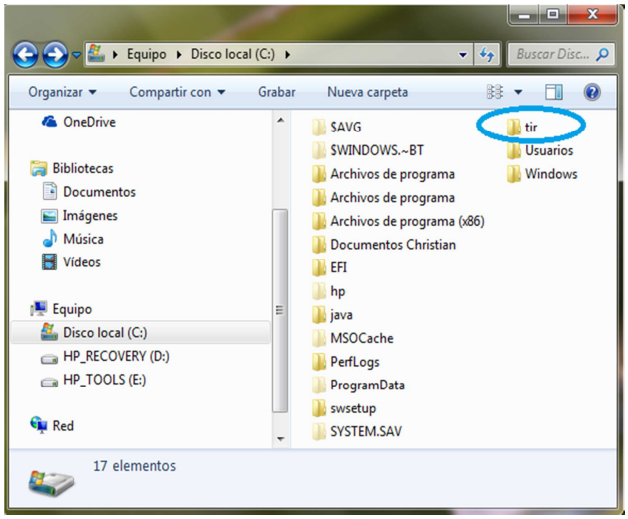
public Integer get_yd(){}

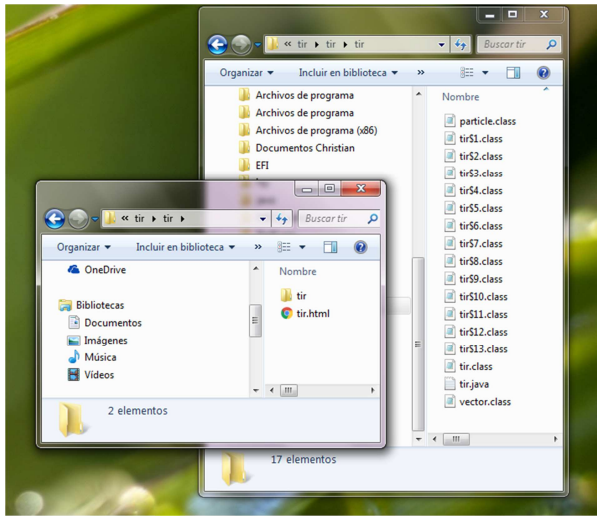


7.6. Implementació del codi en un servidor

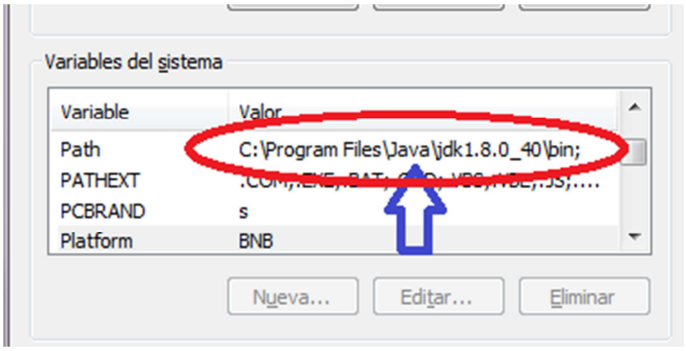
Inicialment anem a executar el codi de forma independent a la plataforma eclipse. A partir de la terminal de Windows. Per això:

a. Col·loquem el java en extensió .class amb totes les classes en una carpeta.

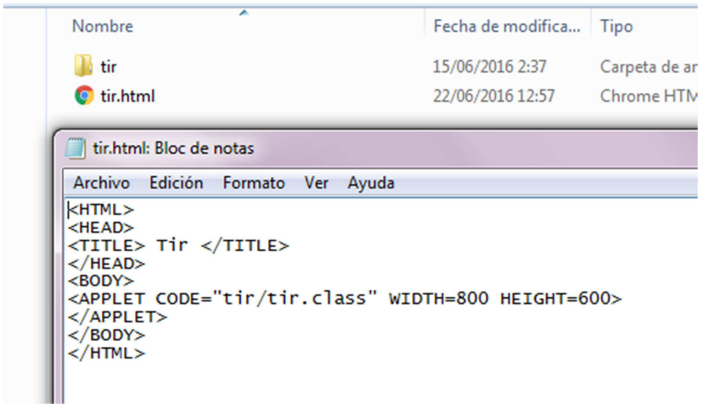




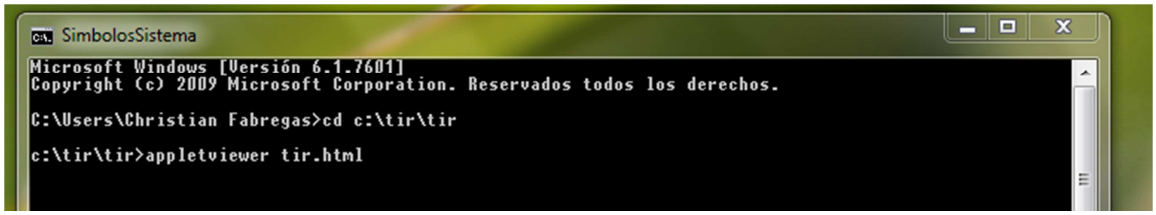
b. Posicionem la “variable d’entorn” del sistema en la direcció en que apunti al java que tenim, em particular ‘jdk1.8.0_40’, de la següent forma:

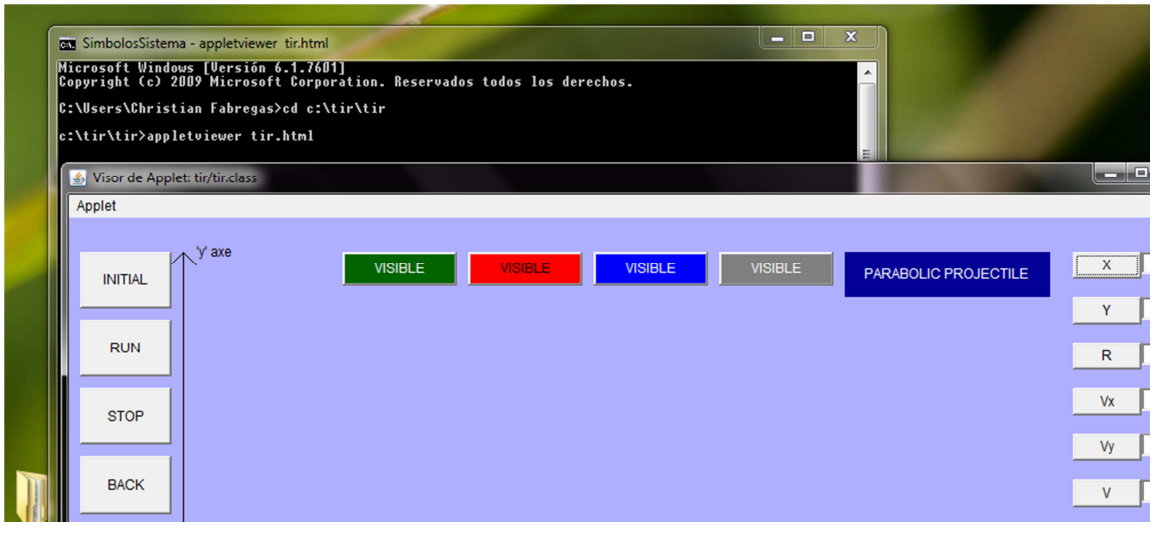


c. Incrustaríem el JAVA en html segons les següents comandes d’HTML:

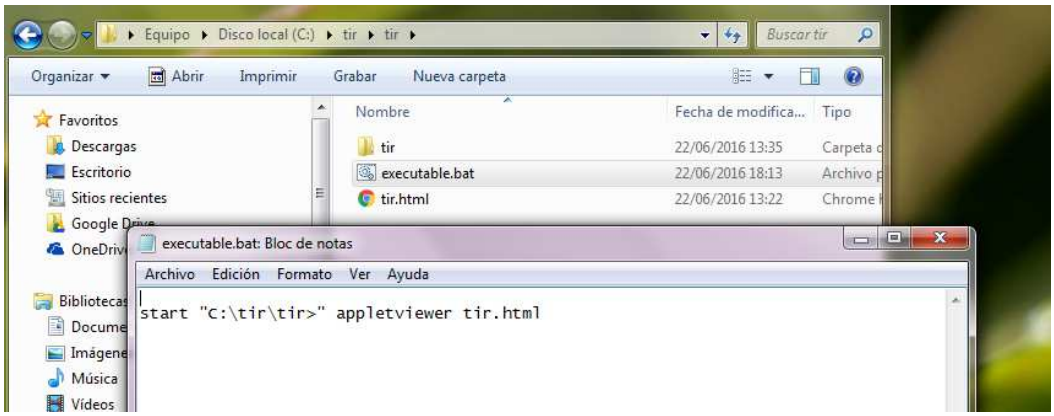


d. Executem des de la terminal en la comanda ‘appletviewer tir.html





e. Creem un .bat per tenir una direcció directe amb un botó sense haver d'accedir a la terminal



f. Finalment havent creat un .bat amb la direcció que obre el programa, es col·locar la carpeta en un SERVIDOR i amb la direcció adequada, s'hauria de poder obrir el programa remotament.

8. Conclusions

a. Ha estat un treball de gran dificultat amb un nombre d'hores considerable. La dificultat fonamental ha estat aprendre un llenguatge de programació visual simultàniament amb la producció del software, la qual cosa ha produït una lentitud notable de la creació del programari.

b. La idea que es desprèn de la finalització d'aquet applet, podria ser projectar en el temps el propòsit de crear més programari, anant més enllà i continuant fent nous applets de la gran varietat de branques de la física que poden ser simulades per ordinador.

c. La creació de software a nivell visual es francament difícil ja que cada detall que passa desapercebut per l'usuari requereix un nombre d'hores enorme per tal que la simulació funcioni correctament.

d. He hagut de buscar solucions en situacions dramàtiques de gran dificultat. On el programa no responia correctament. Especialment en les vectors i les fletxes que esdevenien valors infinits. No ha estat fàcil corregir i controlar els infinits matemàtics que apareixien.

e. Ha estat també especialment difícil controlar els botons que l'usuari utilitza per manipular la simulació. Poden aturar-la, fer-la avançar, retrocedir, esborrar, tornar a començar o interactuar amb les dades d'entrada. La combinació de booleans no ha estat tasca fàcil. En aquest sentit crec que he estat ambiciós volent crear un programa que obté un alt nivell d'interacció amb l'usuari.

f. Es podrien fer millores del programari, tal com:
   1. Obtenir els rectangles vectorials de els 3 grups de 3 vectors: posició, velocitat, acceleració.
   2. Obtenir els radis de curvatura en cada posició
   3. Obtenir les velocitats angular i les acceleracions angulars
   4. Dissenyar simultàniament mes d'un tir
   5. Dissenyar el xoc en l'espai de diferents tirs.

g. En definitiva crec que ha estat un treball molt gratificant i molt edificant. Així com un aprenentatge tant o més intens que qualsevol de les assignatures anteriorment cursades.