Curs: 2020/2021

LA SIMULACIÓ, EL FUTUR QUE JA ÉS AQUÍ!

Prototip de velocitat viari

Tutor: Pepito | Alumne: Katherine Johnson

AGRAÏMENTS

Per començar m'agradaria donar-li les gràcies al meu tutor del Treball de Recerca, per haver-me ajudat tant, per haver estat disponible per a totes les meves inquietuds, dubtes i problemes, sempre que ho he necessitat, i per haver-me proporcionat tot el material, i en general, tot el que em calia per poder dur a terme el meu treball de la millor manera possible. Ha sigut un factor molt important d'aquest treball i sense ell els resultats obtinguts no haguessin sigut els mateixos. Ha fet d'aquest camí tan pedregós, una via més transitable. I per tot això, i segurament per alguna cosa més que em deixo, estic molt agraïda.

També m'agradaria donar-li les gràcies al David, per haver-me donat tant de suport al llarg de tots els mesos envers el meu treball. I també per haver-me ajudat quan no me'n sortia del tot o necessitava algun consell.

Finalment, vull donar les gràcies a tota la gent que ha llegit el meu treball quan li he demanat per donar-me les seves opinions i diferents punts de vista. Arran d'això he pogut veure si anava ben encaminada i què havia de canviar per tal de millorar al màxim.

ABSTRACT

En este trabajo se busca crear una simulación capaz de realizar las operaciones de los problemas de física que se estudian en clase.

Tomando como base los descubrimientos llevados a cabo a lo largo de la historia, se puede aprender como hacerlo.

En el marco teórico, se explica todo el que se tiene que saber para hacer simulaciones, puesto que detrás de estas, hay un gran desarrollo informático.

Finalmente, en el trabajo práctico se comprobará si de los conocimientos logrados sale la simulación esperada.

This work seeks to create a simulation capable of performing the operations of the physics problems studied in class.

Based on the discoveries made throughout history, it is possible to learn how to do it.

In the theoretical framework, it is explained all that is necessary to know to make simulations, since behind these, there is a great computer development.

Finally, in the practical work it will be checked if the knowledge achieved leads to the expected simulation.

SUMARI

RESUM....4

INTRODUCCIÓ....5

Alguarismi....5

Algoritme a l'època medieval....6

Pioners....8

Història de la programació....13

Hardware i software....16

Sistema operatiu....20

Història de la simulació....21

Exemples de simulacions....24

Com es fan les simulacions?....28

Sensors....30

Com es crea un programa informàtic?....32

Què és un compilador?....33

MARC TEÒRIC....34

Els llenguatges de programació....34

Llenguatges de programació més populars....36

Llenguatges que utilitzaré....42

TREBALL PRÀCTIC....43

Part física....43

Programació....45

CONCLUSIONS....53

BIBLIOGRAFIA....55

ANNEXOS....70

RESUM

El somni de qualsevol estudiant és poder resoldre problemes sense necessitat d'haver de fer els càlculs. Vaig pensar que una simulació era exactament això. Amb una simulació podria representar un problema de física i que les operacions les fes l'ordinador. Així doncs, jo només hauria d'introduir les dades i observar què succeïa com per art de màgia. Doncs la realitat és que no és màgia, però gairebé.

Primer de tot vaig haver d'informar-me sobre algoritmes, com van sorgir, i en què van desenvolupar-se. A partir d'aquí vaig anar observant un munt de descobriments magnífics i molt avançats en unes èpoques que ja ens queden bastant lluny. Vaig aprendre molts fets i vaig veure com cada vegada sorgien invents més impressionants. Tot i que el moment amb més avanços va ser en el segle passat, sense les aportacions dels anteriors científics i informàtics, probablement en l'actualitat no sabríem què és un *Iphone*.

La realitat és que cada dia ens trobem simulacions en les nostres vides, com per exemple els videojocs.

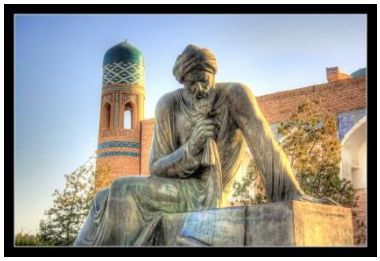
Un cop vaig saber tot el que em calia sobre història, em vaig centrar en la programació, vaig triar els llenguatges que hauria d'utilitzar i em vaig posar a construir la meva pròpia simulació sobre què li passa a un cotxe quan agafem la corba de la rotonda a una velocitat massa alta.

Vaig aprendre tot el que necessitava saber de programació i vaig practicar molt per poder realitzar el meu treball pràctic.

INTRODUCCIÓ

ALGUARISMI

Abu Abdallah Muḥammad ibn Mūsā al-Jwārizmī, un home persa nascut entre els anys 780 i 850, i comunament conegut aquí com Al-Juarismi, va ser qui va aconseguir passar dels números romans als números indo-aràbics.



Il·lustració 1: Estàtua Al-Juarismi

Ell va escriure un llibre anomenat "*Libro de la suma y de la resta, según el cálculo indio*", en el qual explicava que es podia representar qualsevol número utilitzant només 10 símbols, que són els dígits 0,1, 2, 3, 4, 5, 6, 7, 8 i 9 . Així doncs, ell i uns companys van crear, a més, el punt decimal que és aquell que separa la part entera de la decimal.

Combinant geometria amb aritmètica, van crear un nou moviment matemàtic anomenat actualment àlgebra.

Un text escrit al voltant de l'any 825 anomenat en el segle XII "*Algoritmi de numero Indorum"*, té com a traducció "*Algoritmi sobre los números de los indios"* i tracta sobre aquest nou sistema de numeració indo-aràbic. La paraula *"Algoritmi*", si ens hi fixem, prové del nom del seu autor, Al-Juarismi, i actualment l'anomenem algoritme. Però no només ha sorgit una paraula d'Al-Juarismi, *"Guarismo"*, vol dir xifra. El concepte d'algoritme va anar agafant forma i es va anar vinculant a les normes que s'han de seguir per poder calcular amb aquells números que havien descobert. Semblant és, també, el concepte que tenim actualment. Per exemple, quan comencem a aprendre a dividir quan som petits, ho fem seguint una sèrie de passes que ens ensenyen, i no sabem per què però així aconseguim resoldre la divisió. Doncs aquestes passes que seguim són el que anomenem com a algoritme. A mesura que s'ha anat avançant des d'aquests inicis amb Al-Juarismi, aquest concepte s'ha anat ampliant fins al punt en què diem que serveix per determinar un procediment que ens ajuda a resoldre un problema que no té cap altra metodologia.

ALGORITME A L'ÈPOCA MEDIEVAL

Al-Juarismi, a part del text "*Algoritmi de numero Indorum"*, també va escriure un anomenat "*Hisab al-jabr wal-muqabala"*, que traduït vol dir "*ciència de reducció i confrontació"*. Com hem vist abans, en el primer parla de com va introduir un nou sistema de numeració que va ajudar a realitzar uns procediments per resoldre càlculs aritmètics, com la suma o la resta, que avui dia anomenem algoritmes. En el segon, parla sobre equacions lineals, és a dir, sense exponents ni arrels quadrades ni res per l'estil, i també parla sobre equacions quadràtiques que són les que sí que tenen exponent com les de quart grau o segon grau. En comptes de fer-ho enfocat a la geometria, ho feia a partir d'una altra "eina" matemàtica que no era coneguda llavors, i que va anomenar *"aljabr",* o com ho coneixem nosaltres; àlgebra.

Però comencem una mica més enrere en la història, per aproximadament el 2500 aC. En aquella època, a Sumèria, on és Bagdad actualment, s'hi va trobar el primer algoritme gravat en taules d'argila amb caràcters cuneïformes. Ells l'utilitzaven per a mesurar la collita que s'havia de repartir d'una manera justa entre tots els homes que hi hagués. Tot i això, com hem dit abans, la paraula algoritme prové del gran matemàtic Al-Juarismi.

Els algoritmes els podem expressar de diferents maneres segons el context, com per exemple, en els llenguatges de programació o en el llenguatge natural, que és el que utilitzem normalment per comunicar-nos.

En el nostre dia a dia ens trobem sovint amb els algoritmes, com per exemple quan ens posem a cuinar i seguim una recepta, o quan volem anar a un lloc i triem per quin camí anirem, o com he dit abans, quan comencem l'escola de ben petits i aprenem a dividir, o a multiplicar…

El primer algoritme que es va idear per fer funcionar una màquina va ser obra d'Ada Lovelace el 1842. Servia per fer càlculs en la màquina analítica de Charles Babbage. Tot i aquest primer algoritme, als matemàtics de llavors encara els faltava una peça crucial per fer funcionar correctament les màquines. A finals dels anys 30, Claude Shannon va combinar el sistema de càlcul binari que va fer Leibniz, que comptava només amb dos números l'1 i el

0, amb els operadors Booleans, que són les eines de càlcul representades amb els termes “i”, “o”, i “no”, com per exemple:

1	I	1	Dona	1
1	I	0	Dona	0
0	i	1	Dona	0
0	I	0	Dona	0

Aquest operador requereix que dues premisses siguin certes per tal d’obtenir un resultat correcte. Donat el cas en que una de les dues premisses és falsa, no obtenim la premissa resultant.

1	o	1	Dona	1
1	o	0	Dona	1
0	o	1	Dona	1
0	o	0	Dona	0

Aquest en canvi, només requereix que una de les premisses sigui certa per tal d’obtenir un resultat correcte.

No 1=0

No 0=1

I aquest operador, equival a la negació.

Podem observar, doncs, que les taules de la veritat que s’utilitzen a la filosofia, estan relacionades amb aquesta analogia matemàtica.

Un exemple de taules de la veritat en podria ser el següent:

Premissa A: El dofí és blau

Premissa B: El dofí és un cavall

Premissa C: Un cavall és blau

Per tant: A^B=C // NO A^B=NO C (^=i)

Aquest va ser un gran avanç, ja que permetia resoldre gairebé tots els problemes lògic-matemàtics que es plantejaven, i també emmagatzemar dades i editar tant textos com imatges. I va ser en aquest punt quan va començar a sorgir un algoritme més avançat, i també unes màquines capaces de fer la feina de les persones i més encara, simplificar-la.

En aquest camp, si volem que la màquina funcioni sense errors, els algoritmes han de ser ben precisos, indicar un ordre de realització i tenir un nombre establert de passes. Aquesta ha de ser saber identificar les ordres per donar una resposta, sinó dona error. Per això, abans de començar un algoritme s'ha de definir bé el problema, el resultat que esperes, també les indicacions que necessitaràs amb totes les dades corresponents, i establir de manera unívoca les premisses i les implicacions d'aquestes.

PIONERS

Blaise Pascal, va néixer el 1623 i va ser qui va construir la primera màquina sumadora, que era una espècie de calculadora que permetia solucionar problemes amb números. Estava constituïda per un sistema d'engranatges que tenien les xifres del 0 al 9. Quan van nomenar al seu pare comissari real, es va traslladar a Ruán i allà, va començar a pensar en el disseny i la construcció d'una màquina analítica que ajudés el seu pare a la feina. Va anomenar la màquina *"Pascaline".* Aquesta podia fer sumes i restes movent unes rodes petites fetes de metall que es trobaven a la part del davant, i les solucions es veien en unes finestres que estaven a la part superior. Va servir com a model de les calculadores mecàniques i encara se'n conserva algun exemplar.

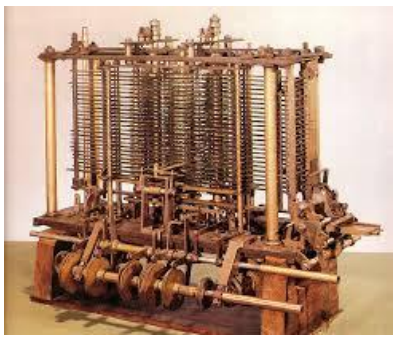
Il·lustració 2: Màquina sumadora

El triangle aritmètic de Pascal, el que dona peu al desenvolupament de binomis amb potències més complexes, dita per ell una "geometria de l'atzar", el va

convertir en un dels pioners del càlcul de la probabilitat, cosa que va ajudar al desenvolupament d'una matemàtica més complexa.

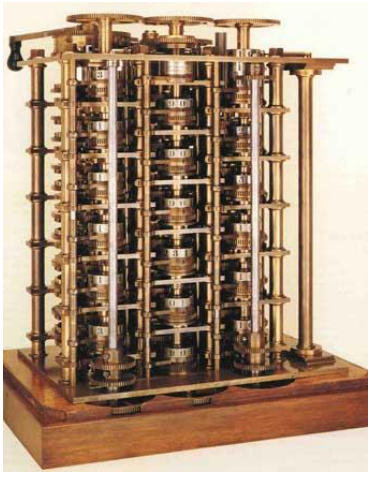
Gottfried Wilhelm Leibniz, nascut el 1646 va inventar i construir una màquina aritmètica que realitzava totes les operacions lineals i també podia calcular arrels quadrades. Era una evolució de la seva antecessora creada per Pascal, per això la va anomenar "calculadora seqüencial o per passes". El 1672, va haver d'anar a París per una missió diplomàtica encarregada pel govern britànic, i tot i que va fallar, s'hi va quedar durant 5 anys. Allà, hi va inventar una altra màquina que podia realitzar unes funcions semblants a l'anterior i, a més va poder elaborar les bases del càlcul infinitesimal, que és l'estudi del canvi. Aquest càlcul constava d'àlgebra, trigonometria i geometria analítica i també era constituït pel càlcul diferencial, les derivades, i el càlcul integral, les integrals.

Charles Babbage, nascut el 1791 va dissenyar el primer ordinador automàtic batejat com a màquina analítica. Ell va treballar en la investigació d'operacions, i més concretament en aquest ordinador, hi havia una combinació entre operacions aritmètiques bàsiques i processos de decisió que consten de 6 parts; la identificació del problema, l'anàlisi d'aquest, la cerca de solucions, la tria de la solució més adient, l'execució d'aquesta, i finalment, la revisió dels resultats un cop aplicada la solució.



Il·lustració 4: Màquina analítica

El 1822 va crear un model petit de la seva màquina diferencial o *"difference engine"*. El seu funcionament aritmètic no era massa complex, era més aviat limitat, però tot i això podia recollir i imprimir taules matemàtiques amb el mínim d'intervenció de les persones possible, que giraven les manetes de la part superior del prototip. Ell també va dissenyar la seva màquina analítica o *"Analytical engine"*. Aquesta contenia totes les parts d'un ordinador



Il·lustració 3: Màquina diferencial

modern, com bé el dispositiu d'entrada, la memòria, la CPU i la impressora. Aquesta màquina és considerada el prototip de l'ordinador modern, tot i que no se'n va arribar a fer un a escala real. Tot i això, si així hagués estat, aquesta hauria hagut d'estar funcionant a base d'una màquina de vapor, i per tant, a causa dels seus components mecànics, no hagués tingut una velocitat de càlcul gaire gran.

Junt amb Ada Lovelace va dedicar tots els seus recursos i tot el que li quedava de vida a crear una màquina capaç de predir els guanyadors de les carreres de cavalls

Augusta Ada Byron, més comunament coneguda com a Ada Lovelace, nascuda el 1815, va ser la primera programadora, ningú abans d'ella havia programat mai. Com hem vist abans, ella va treballar junt amb Babagge en el disseny i la creació de màquines, més concretament, l'analítica. Però, a més, va ser ella qui va desenvolupar teòricament el primer programa que la màquina analítica va utilitzar. El Departament de Defensa dels Estats Units va anomenar a un llenguatge de programació d'ordinadors d'alt nivell ADA, en honor a ella.

Norbert Wiener, nascut el 1894, es va interessar en què els ordinadors de l'època anessin més ràpid, i que en general milloressin. Va desenvolupar les bases de la matemàtica i va determinar que l'única manera possible de resoldre aquest problema seria entendre com funcionen els sistemes de xarxa, que són aquells que es troben interconnectats.

El concepte de "Cibernètica" va sorgir el 1942 amb Wiener com a fundador quan ell treballava en l'estudi de la relació entre màquines i persones, i va elaborar els principis base d'aquesta. Eren, bàsicament, que la cibernètica és la ciència que engloba la comunicació entre éssers vius, màquines, etc, i tracta i estudia les relacions entre aquests. Actualment aquest concepte s'ha aplicat en un àmbit més general. El 1948, va publicar el llibre *"Cybernetics, or control and communication in the animal and the machine"* traduït com a *"Cibernética, o control y comunicación en el animal y en la máquina".* Per poder entendre al màxim la cibernètica, ell va estudiar filosofia i neurologia. Gràcies a això va obrir un mar de noves possibilitats tecnològiques en les quals es podia dur a terme un avanç en la construcció màquines autòmats, és a dir, capaces d'imitar

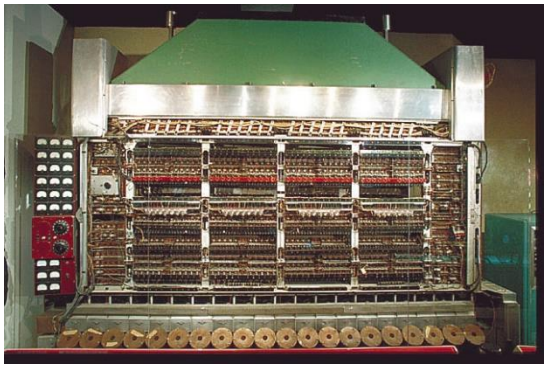
el comportament humà i que funcionen per si mateixes. Això va produir que es pogués desenvolupar molt la informàtica

Howard H. Aiken, nascut el 1900 va construir un ordinador electromagnètic programable. Aquest, funcionava amb unes targetes amb petits forats que li permetien fer operacions. Les targetes contenien les dades que la màquina necessitava, i aquesta les mostrava donant respostes. Algunes persones, consideren que aquest va ser el primer ordinador de la història.

Ell va ser el constructor de la primera calculadora electrònica més moderna, el 1944, i la va anomenar Mark I. Tot i que fos més moderna, no és com qualsevol calculadora actual que puguem imaginar. Aquesta mesurava 15'3 metres de llargada i 2'4 metres d'alçada, a més contenia 800 km de cablejat i aproximadament 3 milions de connexions elèctriques. Tot això per poder realitzar únicament cinc operacions matemàtiques; suma, resta, multiplicació, divisió i aplicant la utilització del resultat anterior. Aquestes les duia a terme mitjançant càlculs amb xifres d'un màxim de 23 dígits i es podien fer gràcies a les targetes amb forats.

Més tard, el 1947, va construir un model millorat que era completament electrònic i el va anomenar Mark II. Per aconseguir fer tot això va estudiar i va fer molta recerca sobre electrònica i tractament de dades, fet que l'ha convertit en un dels pioners de la informàtica.

John Von Newmann, va néixer el 1903 i va ser dels primers que aplicava l'aritmètica binaria, que són les operacions aritmètiques amb codi binari (1 i 0), en un ordinador electrònic. Ell va ser el dissenyador d'un ordinador amb cinta magnètica, que és on s'emmagatzemen dades, ell estava convençut de què les dades i els programes s'havien de poder emmagatzemar en una memòria.



Il·lustració 5: IAS machine

Quan va començar la Segona Guerra Mundial, Newmann va posar-se a treballar amb el govern d'Estats Units. Aproximadament el 1943, va anar desenvolupant interès pels temes de la computació per així facilitar-se la feina. En aquella època ja hi havia molts

projectes d'ordinadors, com els que hem vist anteriorment, i alguns que encara no; Mark I de Howard Aiken o Complex Computers de George Stibiz. Però, ell, concretament va treballar en la ENIAC acompanyat de John Presper Eckert i John W. Mauchly. Més endavant va estar construint la "IAS machine", des del 1946 fins el 1951. Aquest ordinador ha sigut clau per a l'evolució dels ordinadors tal com els coneixem avui dia.

Ell, a més, va fer altres aportacions com per exemple l'ús de monitors per visualitzar les dades i les imatges que transmet, i també el diagrama de fluxe, que és un diagrama que serveix per saber la manera de funcionar de l'ordinador, en aquest cas, i ajuda a ordenar lògicament els elements per estructurar-lo.

La IAS machine, funcionava mitjançant 1000 posicions d'emmagatzemament, que s'anomenem paraules, de 40 dígits binaris cadascuna.

Konrad Zuse, que va néixer el 1910 va ser qui va construir el primer ordinador electromecànic programable. Mitjançant cinta perforada, semblant a la magnètica, va incloure el control programat, és a dir, instruccions definides que es queden programades al dispositiu. Aquest fet va fer que s'automatitzés el procés del càlcul. L'ordinador constava d'una CPU, i una memòria que contenia uns 2600 relés, que són uns interruptors electromagnètics que introdueixen en els ordinadors els números representats de manera binària.

Tot això ho va originar el 1936 i li va atorgar el nom inicial de V1, que més endavant va redenominar com a Z1. Aquest prototip era molt poc avançat. En comptes de transmissors, unes plaques integrades a la placa base, que és on es connecten tots els components, utilitzava unes làmines de metall fines. A part, també estava format per una unitat elèctrica que tenia un motor petit, i la seva memòria constava de 64 paraules que tenien 22 bits cadascuna i estaven en blocs. Actualment un model d'aquest ordinador està exposat en un museu. El 1938 estava preparada per funcionar tot i que no era gaire eficient.

Il·lustració 6: Z1

Per tal de millorar això, va dissenyar un altre prototip anomenat Z2. Era semblant a l'altra però aquesta sí que tenia transmissors, aproximadament uns 800. Per aconseguir-los, va anar a buscar-los a companyies telefòniques que ja no els utilitzessin. Gràcies a aquests, va poder veure que el prototip millorava molt.

Poc abans de la Segona Guerra Mundial, es va posar a treballar en un altre ordinador, que va anomenar Z3 i el va finalitzar el 1941. Després d'això, ell volia seguir millorant els seus projectes i va començar idear el Z4, però, desgraciadament, la guerra va destruir els seus primer ordinadors; el Z1, el Z2 i el Z3. Tot i que va perdre documents i fotografies, anys després va aconseguir reconstruir el Z3 i actualment també es troba exposat a un museu.

HISTÒRIA DE LA PROGRAMACIÓ

Com hem vist, les targetes amb forats van ser un gran avanç per la informàtica, i va ser Ada Lovelace qui va proposar la seva utilització a la màquina de Babagge.

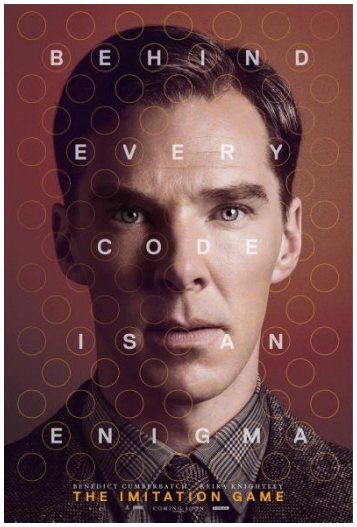
Il·lustració 7: Màquina de l'època

El 1880, a Estats Units, el cens de la població va trigar aproximadament 7 anys a fer-se, per tant, segurament hi havia molt error de càlcul. Herman Hollerit nascut el 1860, qui treballava a una oficina del cens, va crear un sistema per avançar aquest procés. Utilitzant targetes amb forats i circuits elèctrics que permetia llegir 60 targetes cada minut, va aconseguir fer el cens en tan sols 3 anys, fet que va aproximar la xifra a la realitat, va estalviar diners i també temps. Uns anys després va convertir-se en el fundador de la "*Tabulating Machine Company"*, i més tard va sorgir la *"International Business Machines"*, més coneguda com a IBM.

En l'actualitat, els ordinadors es basen en el sistema binari. Aquest s'introdueix en els dispositius electrònics que poden permetre passar el corrent, o no, i per això s'obtenen els dos estats; 0 i 1.

Quan es van començar a construir els primers ordinadors, per introduir aquests 2 estats, s'utilitzaven tubs de buit que són uns dispositius que formen part d'un circuit elèctric que serveixen per modificar, ampliar o substituir un senyal elèctric mitjançant el moviment controlat d'electrons en un espai buit on la pressió és molt baixa.

Alan Mathison Turing, sobre el qual es va fer una pel·lícula anomenada "*The imitation game"*, traduït com a "*El código enigma"*, nascut el 1912, va ser el dissenyador d'una calculadora universal que tenia la funció de resoldre qualsevol problema, l'anomenada "màquina de Turing". Ell va contribuir molt en el desenvolupament de les matemàtiques, entre altres. El 1937 va començar a investigar la lògica matemàtica i el 1943 va aconseguir transformar els seus pensaments en un ordinador que funcionava a base de tubs de buit. George Boole un altre matemàtic nascut el 1815 també va fer les seves aportacions, en concret a l'àlgebra binaria, que va passar a ser "l'àlgebra de Boole", que serveix per simplificar els circuits de l'electrònica digital, i també als sistemes de circuits de l'ordinador.



Il·lustració 8: Pel·lícula "The imitation game"

John Vincent Atanasoff, va néixer el 1903 i va ser el creador del primer ordinador digital electrònic patentat, juntament amb Clifford Berry, nascut el 1918. Això va ser possible gràcies a les obres de Pascal i Babagge, que van ser el seu punt de referència. Així doncs van construir entre els anys 1937 i 1942 l'*"Atanasoff Berry Computer"* o més conegut com a ABC. Aquest era una calculadora electrònica que funcionava amb tubs de buit i que es regia sota el sistema binari.

A la Moore School de la Universitat de Pensilvania va sorgir un projecte per dur a terme unes taules de tir que proporcionaven les dades necessàries per saber com disparar una arma balística, dins d'unes condicions climatològiques, amb precisió perquè arribi a l'objectiu. Fer aquests càlculs a mà era molt complicat

perquè es tractava de càlculs molt complexes que duraven setmanes. John Mauchly, nascut el 1907, era l'encarregat de dirigir el departament de física del "Ursine College" de Filadèlfia. Casualment, va viure durant 4 dies a la casa d'Atanasoff, fet que li va permetre conèixer una mica els seus pensaments i les seves idees. Com hem vist abans, ell i John Presper Eckert, nascut el 1919, van crear un ordinador electrònic operacional a gran escala per tal que pogués accelerar els càlculs del projecte de la Universitat de Pensilvania. Aquest és l'anteriorment anomenat "*Electronic Numerical Integrator And Computer"* o ENIAC. L'ENIAC, estava compost per 18.000 tubs de buit en un espai de 84 metres cúbics. El seu pes era d'aproximadament 30 tones i consumia uns 100.000 watts. Podia calcular fins a 5.000 operacions cada segon, tot i que algú havia de programar-lo connectant-lo a 3 taulers elèctrics que tenien més de 6.000 interruptors. A vegades per fer això podien passar-s'hi dies.

D'altra banda, el primer llenguatge de programació que no s'havia de programar en codi binari i que era universal perquè es pogués utilitzar en qualsevol ordinador, va ser el *"Common Business-Oriented Language"* traduit com a "Lenguaje Común Orientado a *Negocios*" i més conegut com a COBOL. Grace Murray Hopper, nascuda el 1906, l'any 1952 va crear el primer compilador, que era un programa que podia passar ordres en un idioma humà al codi binari de l'ordinador, fet que va ser una gran fita per a l'avanç de la programació i la informàtica en general.

Més endavant, el 1958, es van deixar de fer servir tubs de buit per utilitzar transistors. Aquests van ser un gran avanç, ja que permetien que les màquines es poguessin construir a una mida més reduïda, ja que el seu tamany també era més petit al dels tubs de buit. Aquell mateix any, Jack Kilby va desenvolupar el circuit integrat o xip de mà, substituint els tubs de buit. El 1961, es va dur a terme el procés d'introduir el circuit elèctric en un xip de silici. L'any 1971 va sorgir el primer microprocessador disponible al comerç, que és un circuit integrat més complex, anomenat Intel 4004, de 4 bits. Dos anys després, el 1973, Gary Kildall va crear el sistema operatiu CP/M o *"Control Program for Microcomputer"* que funcionava amb microprocessadors de 8 bits. Amb tot això, l'any 1980, IBM va llençar al mercat el primer ordinador PC, que vol dir

literalment *"Personal Computer"* o "ordinador personal". El 1998 la IBM va acabar el projecte "*Blue Pacific"* que es tractava de l'ordinador més potent fins llavors. Estava compost per 5.856 processadors (CPU) que tenien una velocitat de 3'9 Teraflops (1 flop és la quantitat d'operacions que es poden fer en un segon) i 2'6TB de memòria. A més, feia 2.400 metres quadrats i pesava 47 tones. Es feia servir per simular explosions nuclears i va costar aproximadament 13.000 milions de pessetes que serien uns 80.000 milions d'euros.

HARDWARE I SOFTWARE

El hardware són tots els components físics requerits per fer funcionar l'ordinador i tot el que aquest engloba. Dins d'aquest hi consten, entre d'altres, la impressora, el ratolí, els discs durs i les memòries.

L'evolució del harware està dividida en 4 etapes al llarg de la seva existència; del 1945 al 1956, es van començar a utilitzar els anteriorment anomenats tubs de buit, del 1957 al 1963, va ser quan es van substituir per transistors, del 1964 a l'actualitat, és quan es va passar a utilitzar circuits integrats o elèctrics en xips de Silici, i el que s'està estudiant i dissenyant pel futur és un hardware fet amb altres materials diferents a aquest.

Altres elements del hardware són les targetes foradades i també hi ha altres elements que es van introduir més endavant com els monitors.

Hi ha diferents tipus de hardware:

- Hardware de processament

Aquest va lligat a la CPU (Unitat Central de Processament), que és on es duen a terme totes les operacions de l'ordinador, allà les interpreta i posteriorment executa les ordres necessàries perquè la resta de components de l'ordinador funcionin correctament.

Il·lustració 9: CPU

- Hardware d'emmagatzemament

Aquest són els components que s'encarreguen d'emmagatzemar tota la informació perquè siguin de fàcil accés. El component més destacat és la memòria RAM (Random Access Memory), però hi ha d'altres com per exemple els discs durs, els USB (Universal Serial Bus) o les SSD (Solid-State Drive).

Il·lustració 10: USB

- Hardware gràfic

Com bé indica el seu nom és aquell que s'encarrega d'interpretar i executar les ordres dirigides al muntatge d'imatges. No es duu a terme directament a la CPU, ja que no funcionaria d'una manera tan eficaç, per això els components són targetes gràfiques que tenen memòria i CPU interns, de manera que el sistema funciona d'una manera molt més òptima.

- Dispositius perifèrics

Els perifèrics són un tipus de hardware que fan de via intercomunicadora entre el món fora de l'ordinador i l'interior. Dins d'aquests també hi ha diferents tipus. Els d'entrada són aquells en què la informació passa de fora a dins, i serien, per exemple, el micròfon o el teclat. Els de sortida són aquells en què la informació passa de dins a fora, i serien, per exemple, la impressora o els altaveus. I finalment els d'entrada i sortida o mixtos, que són aquells que duen a terme les dues funcions, un exemple en seria les pantalles tàctils.

Il·lustració 11: Perifèrics

Per una altra banda, tenim les IRQ, que són “Interrupcions de hardware”. En aquestes, el hardware avisa a la CPU mitjançant senyals als circuits d’aquesta. Quan un component de l’ordinador necessita enviar un missatge per demanar alguna petició, ho fa a través de les *“Interrupt Request Line”* traduït com a “Línies d’Interrupció”. Les IRQ, bàsicament fan d’intermediàries entre les peticions dels components de l’ordinador i la CPU, on es durà a terme, o no la petició, seguint un ordre de prioritats. El que farà la CPU és resoldre la petició i tornar a la seva feina constant, que és dur a terme milions d’operacions per segon.

A part del hardware, l’ordinador també requereix d’un software, que és on es troben les ordres codificades que s’han de dur a terme. Són tots els components lògics requerits per fer funcionar un ordinador, és a dir, un programa o un conjunt d’aquests que permet executar totes les ordres del sistema informàtic. La utilització d’ambdues parts dona peu a què l’ordinador pugui desenvolupar-se d’una manera més eficient.

Hi ha diferents tipus de software:

- Sistema operatiu explicat posteriorment, i principal software per fer funcionar un ordinador.
- Software de sistema

És el programa que s’encarrega d’executar totes les aplicacions que calen perquè el sistema funcioni de la manera adequada. És diferent del sistema operatiu, ja que consta a més, d’eines per obtenir una optimització de funcionament màxima, components que controlen el dispositiu i els servidors d’aquest.

- Software de programació

És on es troben totes les eines necessàries per a la construcció de nous softwares. Aquest només l’utilitzen els experts programadors. Dins d’aquest hi podem trobar eines com els compiladors, que són programes que interpreten i executen altres programes, els intèrprets, que és el que s’encarrega de traduir els llenguatges de programació, o els editors de text, que s’encarreguen d’analitzar els arxius que contenen text, els llenguatges de programació o els codis font, que són arxius amb instruccions que ha de realitzar la CPU i que

serveixen per fer un programa i compilar-lo, que significa que passa la informació a codi binari perquè l'ordinador l'entengui.

La recerca que he realitzat buscant informació al respecte és molt extensa i difícil de plasmar en un apartat així de manera senzilla. Simplificant-ho extremadament, el software de programació, és el conjunt d'ordres que executarà l'ordinador per obtenir un resultat, a partir d'uns imputs. El programa per a fer la declaració de la renta anomenat "PADRE", està construït amb aquest tipus de software de programació.

- Software d'aplicació

És un programa que ajuda a facilitar algunes feines dels ordinadors, mòbils, etc. Més comunament conegut com a APP o aplicacions.

- Software maliciós o malintencionat

És conegut com a *"Malware"*, i bàsicament s'utilitza per aconseguir informació privada d'algú o simplement malmetre el seu sistema.

Il·lustració 12: Malware

- Software lliure

Aquest són programes que es poden utilitzar de manera gratuïta, el codi font és obert i públic, i deixen a l'usuari per al qual han estat dissenyats, modificar, copiar, utilitzar, etc, els programes i el codi font de manera personal. És a dir, qualsevol persona hi pot accedir, i per tant, pot fer-hi ús i modificar-lo. Un exemple seria el Mozilla Firefox, que ha estat modificat per donar peu al "Tor", el navegador de la "*deep web*", i el "Brave", un navegador que no et rastreja ni agafa informació personal.

Il·lustració 13: Brave

- Software propietari

Aquest vindria a ser el contrari de l'anterior. No deixa a l'abast de tothom el seu ús, per això, en aquest cas no ofereix un codi lliure, sinó que és privat, ja que ha estat creat per una empresa, i per tant, cap usuari el pot modificar ni accedir al codi font. Tot el sistema de Windows, el paquet de Microsoft Office, etc, que

s'han de pagar per utilitzar-los, mitjançant una llicència, en són exemples.

Il·lustració 14: Paquet Microsoft Office

SISTEMA OPERATIU

Un sistema operatiu és aquell programa que fa funcionar l'ordinador. Podríem dir que és l'intermediari entre les aplicacions i la màquina en si mateixa. És el que ho gestiona tot; controla l'execució de programes; automatitza les feines necessàries perquè tot funcioni correctament i de manera que sigui còmode a l'hora d'utilitzar-ho quan, sense ell, s'hauria d'invertir molt de temps en configurar-ho tot amb un terminal, que és un intèrpret d'ordres, enviant comandes, és a dir, peticions; optimitza els recursos del sistema; carrega els perifèrics i si cal algun programa per controlar-los i fer-los servir correctament, l'instal·la; organitza el sistema d'arxius perquè sigui més fàcil treballar amb les carpetes i arxius; detecta errors de programes o maquinària; optimitza els recursos del sistema; gestiona la memòria relacionada amb els programes que s'utilitzen i també les comunicacions en xarxa… Tot això per a que l'usuari ho pugui fer servir d'una manera més senzilla. Aquest pot disposar d'interfície gràfica, que és com acostumem a tenir el nostre ordinador o amb un terminal, que seria amb una pantalla negra i lletres blanques en el que introdueixes les comandes i ell executa les ordres. Per poder dur a terme totes aquestes funcions, s'instal·len drivers, que són controladors de dispositius. Sense aquest, el hardware no es podria utilitzar. També s'instal·len apis, que són uns protocols que es fan servir per desenvolupar i integrar el software en les aplicacions i també permet la comunicació entre aplicacions, llibreries, que contenen codificat el necessari pel programa escollit… Per exemple, si l'usuari vol instal·lar el Google Chrome, el sistema operatiu s'encarrega de fer-ho, determinar que farà servir "x" de memòria RAM i també integrar el ratolí a l'entorn gràfic desitjat per tal que pugui navegar còmodament. El sistema

operatiu també pot ser de codi lliure, i per tant, gratis o de codi privat i per tant amb propietari i de pagament, com el Windows comentat anteriorment.

HISTÒRIA DE LA SIMULACIÓ

Claudio Rocchini Buffon, l'any 1777 va plantejar el problema de *"La aguja de Buffon"*. Aquí va ser on va començar a sorgir la simulació. Aquest problema matemàtic serveix per aproximar el nombre π. Per a fer-ho, es deixa caure una agulla a sobre d'un full amb ratlles, i després, s'observa i s'apunten totes les vegades que l'agulla creua les ratlles. Ell va realitzar aquest experiment moltes vegades i va poder veure que estava relacionat amb el nombre π. Si es llença moltes vegades l'agulla, després es multiplica la quantitat de vegades per 2 i es divideix entre el nombre de vegades que ha creuat les ratlles, s'obté un nombre molt aproximat a π. Pierre Simon Laplace, l'any 1812 va agafar aquest problema i el va millorar corregint les seves mancances. Així doncs, va passar a anomenar-se el problema de "Buffon-Laplace". Ell també va introduir la "Teoria Analítica de les Probabilitats". En aquesta, estudia els principis i les aplicacions de la "geometria a l'atzar", que comentàvem abans. Aquí introdueix els recursos d'un anàlisi matemàtic per a estudiar els fenòmens aleatoris i recull unes memòries que van ser publicades l'any 1771. Ell resumeix el càlcul de probabilitat dient que "en el fons, la teoria de probabilitat és només el sentit comú expressat amb números". També afegeix que "és notable que una ciència que va començar amb consideracions de jocs a l'atzar havia d'arribar a ser l'objecte més important del coneixement humà. Les qüestions més importants de la vida constitueixen en la seva major part, en realitat, només problemes de probabilitat". Després d'ell aquestes qüestions van deixar de ser importants fins que va desaparèixer al segle XIX com a disciplina matemàtica. El 1779, va escriure un document on explicava el mètode d'estimar la proporció entre el nombre de casos favorables i el nombre de casos possibles, a més, també va escriure els conceptes de funció generatriu, que és aquella que permet aplicar eines funcionals als problemes de successions, el principi dels mínims quadrats, la solució al problema de l'agulla…

Arthur Guinness, l'any 1816, fundador de la fàbrica "Guinness", va aplicar en la seva destil·leria i explotació agrícola els seus coneixements sobre estadística.

Il·lustració 15: Guinness

Més tard, William Sealy Gosset, l'any 1899, a partir dels seus coneixements sobre estadística, va introduir l'ús de les simulacions en el procés de control industrial, ja que ell treballava en el departament de fermentació de la fàbrica de Guinness. Per a trobar solucions a problemes de la indústria i l'enginyeria, es va basar en l'experimentació i en les tècniques d'anàlisi.

Als anys 40, van sorgir els primers ordinadors com, per exemple, l'ENIAC, comentat anteriorment. A més també es va posar en pràctica el mètode "MonteCarlo" en els ordinadors moderns, per part d'Stanislaw Ulam i John Von Newmann. Aquest és un mètode de simulació que calcula estadísticament el valor que tindrà una sèrie de successos. Aquesta simulació es realitza en ordinadors amb uns programes específics.

En l'any 1950, mitjançant l'ordinador ENIAC, Newmann, J.G.Charney i R.Fjörtoft van crear el primer model numèric de predicció del temps que va originar resultats positius, aquesta podria ser la primera aplicació de les simulacions en mapes que mostren la predicció meteorològica. En aquella època, a causa de la guerra, era molt important saber el temps que faria determinats dies. Malauradament, quan va sorgir el sistema més efectiu, la guerra ja havia acabat, però s'havien pogut utilitzar altres models més o menys eficients.

Més enllà, en l'any 1957, AMA (*"American Management Association"*) una organització internacional creada el 1923, va crear la primera simulació gerencial. L'objectiu era oferir als seus 100.000 membres i persones, serveis a través de "classes" que es basaven en temes de direcció i gestió d'empreses. El 1961, es va celebrar la "*Conference on Business Games as Teaching Devices"* on es va definir el terme "business *game"* com a "una situació artificialment creada (una simulació) en la que diversos jugadors han de prendre decisions cada cert temps en un entorn de negoci fictici, que poden

afectar a les condicions de l'entorn futur. A més, les interaccions entre les decisions i l'entorn es determinen per un procés d'arbitratge que no està obert als jugadors".

Keith Douglas Tocher, l'any 1960, va aconseguir fer un programa que simulés una planta de producció on les màquines mostraven els estats: ocupat, esperant, no disponible i error. Fins i tot, es va dedicar a explicar com anava tot aquest funcionament en un llibre anomenat "*The art of simulation"*. Entre aquell any i el següent, va néixer el *"General Purpose Simulation System"* traduït com a "Sistema de Simulació de Propòsit General" també anomenat com a GPSS. Aquest era un llenguatge de programació que estava fet per dur a terme simulacions en l'àmbit dels teleprocessos, que serien, per exemple, controlar el trànsit, reservar bitllets de tren…

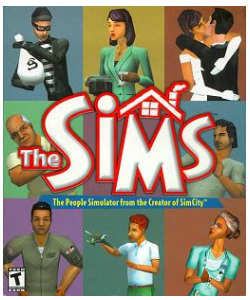
Posteriorment es van desenvolupar altres llenguatges de programació. El *"Norsk Regnesentral"* (NR) o "Norwegian Computer Center", va desenvolupar amb l'UNIVAC (*"UNIVersal Automatic Computer")*, una empresa d'ordinadors, el projecte del programa SIMULA. D'aquí en va sorgir el SIMULA I, creat el 1962 per Dahl i Nygaard, que és un llenguatge de programació. El seu compilador es va integrar l'any 1964 en els ordinadors 1100 d'UNIVAC i més tard també en el Burroughs B5500 i l'URAL 16. El 1963, va sorgir el SIMSCRIPT, un altre llenguatge de programació que utilitza DES (*"Data Encryption Starndard"*), un mètode de simulació amb un algoritme de xifrat del rendiment i de com es comporta el procés, la instal·lació o el sistema real. Aquest és un mètode alternatiu al GPSS que es basa en FORTRAN, acrònim de "***For****mula* ***Tran****slator"*, el llenguatge d'alt nivell de càlcul numèric més antic, creat inicialment pels ordinadors d'IBM, i utilitzat en ciència i enginyeria. També s'utilitza per al desenvolupament de programes que analitzen els superordinadors (ordinadors més potents).

Il·lustració 16: UNIVAC

El WSC que vol dir *"Winter Simulation Conference"*, es va crear el 1967, i és on hi ha registrats des de l'any 1968 fins a l'actualitat tots els llenguatges de simulació, arxius i documents relacionats. Dins la meva recerca, aquest ha estat un gran descobriment que m'ha proporcionat molta informació.

Durant els anys 1970 i 1981, les simulacions van començar a avançar i es van desenvolupar moltes eines per a fer-ho possible. Van sorgir estudis on es deia que les persones podíem interpretar millor els canvis en certes situacions si ho observàvem en gràfiques, imatges o animacions formades a partir de dades. Abans d'això, les dades que sorgien de les simulacions en ordinadors, s'observaven en taules.

A partir d'aquell moment, en els anys següents van anar sorgint simulacions aplicades a diferents camps; l'any 1990, van sorgir les simulacions per als pilots aeris, on podien practicar en simuladors d'avions, l'any 1992, van sorgir les simulacions encarades a l'entreteniment, com per exemple els jocs, el 2000, es va llençar al mercat el videojoc "Els Sims", el qual simula la vida de les persones i no té cap més objectiu que construir una vida, l'any 2008 el CERN, l'Organització Europea per la Investigació Nuclear, va iniciar un projecte que simulava l'inici de l'Univers per investigar sobre les condicions i els elements que hi havia en el moment del Big Bang, el 2013, va començar el projecte finançat per la Unió Europea anomenat *"The Human Brain Project"* (HBP). Aquest es va idear per avançar en la investigació de la neurociència, la medicina i la informàtica. És un simulador fet per superordinador de l'estructura i la funcionalitat del cervell. A més, també s'estudia l'ètica implicada en aquest projecte, etc.



Il·lustració 17: Els Sims

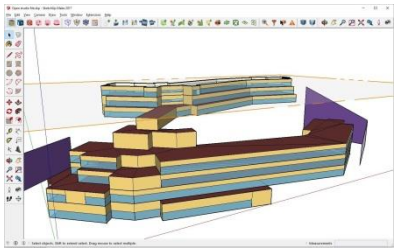
EXEMPLES DE SIMULACIONS

- Com es reparteixen les característiques en les estructures d'un edifici que permet saber si una determinada construcció és viable o no:

Per a fer aquestes simulacions existeixen diversos programes.

Programa OPENSTUDIO:

És un programa gratuït que realitza simulacions a partir d'un model de l'edifici creat inicialment a l'SketchUp, un programa de disseny gràfic en 3D. Després es passa a l'OpenStudio, el qual ofereix les eines adequades per a editar el model i fer la simulació corresponent que permet l'anàlisi del comportament

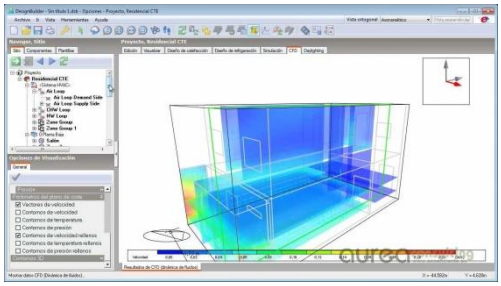


Il·lustració 18: Exemple OS

energètic de l'edifici. Això és útil per a saber com utilitzar els serveis de calefacció, refrigeració, il·luminació i ventilació, i per a fer-ho es fa servir el motor de càlcul, que és l'element que duu a terme tots els càlculs, d'EnergyPlus, un programa de simulació. A més també es poden obtenir dades sobre el clima meteorològic de qualsevol punt del món, amb les que es construeixen diagrames bioclimàtics que permeten idear dissenys adequats al territori.

Programa DESIGNBUILDER:

És un programa que fa simulacions del comportament tèrmic dels edificis per a optimitzar el disseny d'aquests. També ho fa amb el motor de càlcul d'EnergyPlus, amb l'ús d'un generador de models en 3D i una interfície, que és un dispositiu que passa la informació de l'ordinador en informació comprensible per persones, molt senzilla. Mitjançant aquest programa s'obtenen dades sobre la ventilació inicial dels edificis per a poder determinar les característiques més afables per a aquests posteriorment. També pot calcular el consum d'energia i la quantitat de CO_2 que emeten els edificis avaluats. Això permet saber quins sistemes de climatització segons el comportament tèrmic s'han d'utilitzar tenint en compte l'impacte que causen els edificis en el seu entorn.



Il·lustració 19: Exemple DB

Programa ECOTECT:

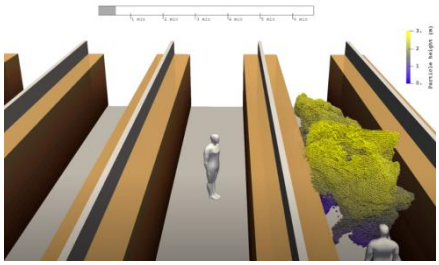
És un software que ofereix realitzar el disseny eficient d'edificis amb una interfície encara més senzilla i intuïtiva que la de DesignBuilder, i permet que el model s'exporti a altres programes de simulació que tenen com a base el motor de càlcul d'EnergyPlus. Aquest també permet obtenir dades meteorològiques de qualsevol punt del món per al seu anàlisi i posterior obtenció de diagrames bioclimàtics per poder idear dissenys i, després veure i calcular qualsevol

element aplicat a l'edifici. Aquest, no consta de motor de càlcul per treballar en la dinàmica de fluids, que serveix per estudiar la ventilació inicial i després l'artificial, etc. Així doncs, per calcular el consum energètic només es pot fer una aproximació, per tant, els resultats són menys precisos que en el DesignBuilder o en l'OpenStudio.

Il·lustració 20: Exemple ET

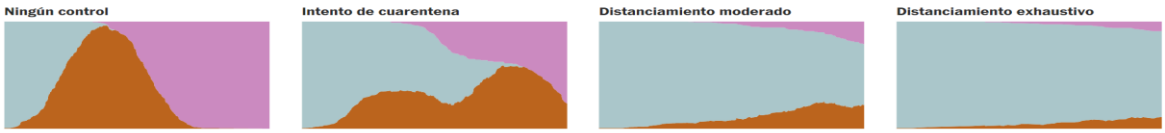
- Simulacions del coronavirus:

Aquesta simulació feta a través de superordinador representa el moviment de les partícules d'aerosol, que són tant sòlides com líquides en un medi gasós, d'uns 20 micròmetres (0'02 mil·límetres) en l'aire. La tos seca que produeix un dels símptomes del coronavirus té una mida d'aproximadament 15 micròmetres. Aquestes partícules de la tos, no cauen a terra sinó que es queden en suspensió a l'aire. Aquesta simulació ajuda a entendre el moviment d'aquestes partícules que no podem percebre amb l'ull humà.



Il·lustració 21: Simulació COVID-19

En la següent simulació, diferent de l'anterior, es mostren diferents escenaris:



Il·lustració 22: Simulació 2 COVID-19

En el primer, es presenta un grup de gent convivint sense cap mena de restricció. Quan hi introdueixen una persona amb el virus, els contagis van augmentant de manera molt ràpida i no passa gaire estona fins que tot el grup està contagiat. Llavors, podem veure com la corba és ben pronunciada i de manera exponencial.

En el segon, es mostra un grup també però amb mesures de distància de seguretat. En aquest cas, el grup triga més a contagiar-se i, per tant, la corba és menys pronunciada i els contagis es troben més separats en el temps.

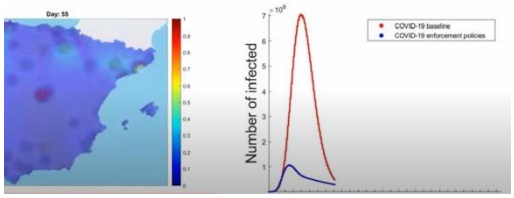
En el tercer, es mostra el grup confinat. Com les persones deixen d'estar en contacte, el contagi deixa de ser tan repetit i, per tant, la corba és més plana i dura més temps. Amb això com més dura el període de contagi entre persones, més dura la corba en el temps, i per tant, es produeixen menys contagis i menys morts.

Aquesta simulació ha estat creada a partir d'un treball publicat per *"The Washington Post"*, el 14 de març de 2020.

Es crea d'acord amb unes dades, com per exemple, que hi ha pacients que es fan immunes al virus sota unes variables que depenen del període d'incubació de cada persona, que cada persona contagiada pot infectar 3 persones i té una probabilitat del 6% de morir, que les persones confinades tenen una probabilitat de contagiar-se del 33%…

I finalment, hi ha un quart escenari on es representa la "nova normalitat", respectant els protocols i mesures de seguretat. D'aquesta han extret que si la gent torna a treballar amb total normalitat, sense tenir en compte les mesures de seguretat, hi ha una gran probabilitat de produir-se un rebrot. Però sinó, és completament sostenible.

En aquesta altra i darrera simulació del simulador EpiGraph, es poden representar les característiques variades dels diferents tipus de població, com estudiants, persones de tercera edat, treballadors, persones en l'atur, etc, les seves relacions en uns determinats entorns com l'escola, les residències, la feina, etc, i un model de transport que mostra la dinàmica espaial de la propagació del virus a diferents llocs. A més, també relaciona la propagació del virus amb condicions meteorològiques com la temperatura, la pressió atmosfèrica i la humitat. Aquest estudi ha estat



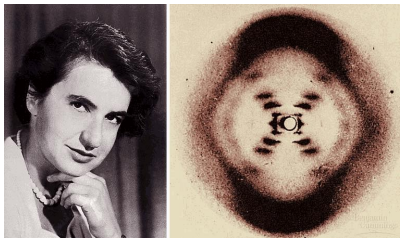
Il·lustració 23: Simulació EpiGraph

liderat pel grup de recerca anomenat ARCOS (Arquitectura de Computadores, Comunicaciones y Sistemas) de la UC3M que és la Universidad Carlos III de Madrid.

COM ES FAN LES SIMULACIONS?

Les simulacions són programes informàtics empleats en format multimèdia, és a dir, que utilitzen diferents mitjans simultàniament per a transmetre una informació. Aquestes permeten la modificació d'uns determinats paràmetres per a observar com afecta en el que s'està treballant. Per fer-les, es creen unes animacions mitjançant programes per l'ordinador, de situacions reals, per tal d'analitzar-les i poder trobar solucions d'una manera més efectiva i més senzilla sense haver de fer primer proves físiques. Per exemple, un software de simulacions bastant conegut és Simulart. Ells van crear Software ProModel, que és per crear simulacions i treballen sota ambient Windows. Algunes aplicacions en aquest format serien; Google Chrome, el Word o l'Excel. Però, en concret el ProModel serveix per simular, analitzar i optimitzar sistemes de tota mena. A més, el seu ús és fàcil però alhora complex per a ser capaç de simular qualsevol situació requerida.

Des dels inicis de la humanitat, l'ésser humà ha buscat explicar els fenòmens que l'envolten a partir de l'observació d'aquests. Arran d'això, van sorgir els models científics, no se sap amb exactitud quan, però al voltant dels anys 50. L'any 1947, l'Alan Turing, va dirigir el Computing Machine Laboratory de Manchester per dissenyar el MADAM, un ordinador que emmagatzemava un programa en la seva memòria principal. Tot i això, abans que aquest, ell mateix va participar en el disseny de la ACE, que tenia la mateixa funció però amb menys capacitat. Gràcies a tots aquests descobriments, va aconseguir desxifrar codis nazis. Un altre exemple en seria la IAS machine, completada el 1952 per Jhon von Newman, un professor de matemàtiques a la Universitat de Princeton i al Institute for Advanced Study (IAS). L'any següent, el 1953, va ser quan Watson i Crick van aconseguir observar



Il·lustració 24: Rosalind Franklin

l'estructura molecular de doble hèlix de l'ADN, a partir de Rosalind Franklin. Per a fer-ho van construir un model científic que contenia tota la informació que tenien sobre l'ADN. Així que podem dir que els models científics són essencials per a la investigació, ja que ens donen respostes sobre fets, fenòmens naturals, els quals representen simplificadament, teories científiques, etc. En general, ens descriuen i expliquen el que passa al nostre món. Tot i això, encara queda molt per avançar en aquest àmbit, i no hi ha respostes a totes les preguntes, sobretot a la de com funciona el propi model. Per a constituir els models, s'utilitzen lleis específiques per a cada problema. Els simuladors de túnels del vent, són una simulació aerodinàmica, en què tot el que envolta vehicle, com bé el consum de combustible, el refredament del motor o el soroll intern a la cabina, s'estudia a partir de la llei de la dinàmica, ja que tot el que cau sobre aquest són forces aerodinàmiques. Per a fer-ho utilitzen una simulació CFD (Computational Fluid Dynamics) que serveix als enginyers per visualitzar, calcular i determinar prèviament la dinàmica del vehicle, abans de fer proves en el túnel del vent físic, per finalment construir-lo. Així doncs, veiem que és una gran eina, molt útil, i no només en aquest camp. Per exemple, per a dur un control dels cultius openfield d'un pagès i facilitar l'estudi dels regs, els períodes de sembra, etc, també s'utilitzen simuladors, com el d'INIA (Institut Nacional d'Investigació Agropecuaria) o bé la FAUBA (Facultat d'Agronomia de la Universitat de Buenos Aires), entre d'altres. També s'utilitzen simuladors per calcular la quota de la hipoteca, proporcionats pels diferents bancs, com Santander, BBVA, Bankia, etc. Així doncs, podem observar que les simulacions, actualment, es troben en molts aspectes de la nostra vida, fins i tot els més quotidians, i per a dur-les a terme, és essencial l'ús d'una llei matemàtica i un model.

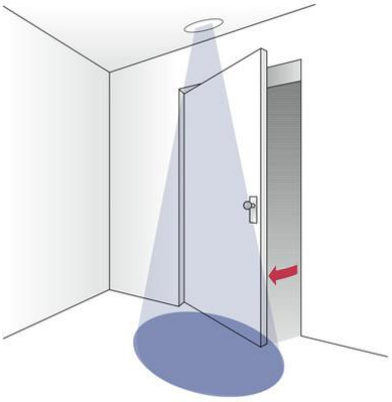
SENSORS

Els sensors són els que passen la informació externa que reben a un impuls elèctric, generalment digital, que permet el seu anàlisi i el seu processament a la Unitat de Control del Sistema, SCU o "*System Control Unit*", que és la que representa la unitat de control central pels equips de processament com els sensors i permet controlar els processos específics mitjançant una programació gràfica. La finalitat dels sensors és proporcionar informació als programes de simulació de manera automàtica.

Existeixen diferents tipus de sensors segons la seva funció:

- De contacte

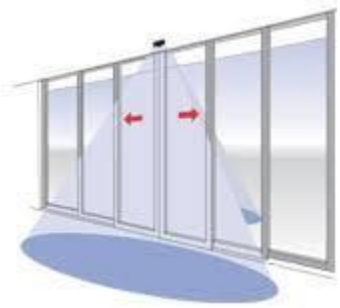
S'utilitzen la posició final dels components mecànics. Els principals són els "fines de carrera" o "finals de carrera". Aquests són interruptors que tenen un peça petita amb mobilitat i una altra sense, és a dir, fixa, anomenada NO ("normalment obert") o NT ("normalment tancat"). Aquests sensors són útils, per exemple, quan és necessari saber quan les portes automàtiques han arribat al seu límit d'obertura o tancament, i llavors, el motor que les fa moure, ha de parar.



Il·lustració 25: Sensor de contacte

- Òptics

Aquests tenen la capacitat de detectar un cos quan aquest interacciona amb el feix de llum connectat al sensor. Els principals en són les LDR o fotoresistències. El valor d'aquestes disminueix amb la llum, així doncs, quan estan en contacte amb un feix de llum,



Il·lustració 26: Sensor òptic

deixen passar el corrent elèctric. En canvi, quan hi ha algun obstacle, com seria un cos que impedeix el pas de la llum, les LDR augmenten la resistència i paren el pas de corrent pel circuit de control.

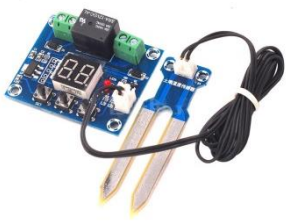
- De temperatura

Els principals són els termistors, aquests són resistències que quan el seu valor augmenta amb la temperatura s'anomenen PTC (*"Positive Temperature Coefficient"*) i quan disminueix NTC (*"Negative Temperature Coefficient"*). Així doncs, depenent de la temperatura que rebi el termistor, permetrà el pas de corrent o no.

Il·lustració 27: Sensor tèrmic

- D'humitat

Aquests tenen en compte que l'aigua és un material conductor de l'electricitat. Així doncs, uns cables sense res més, conduiran molt poc corrent en un ambient humit, en canvi, si hi afegim un transistor, que amplifica aquest corrent, obtindrem un detector d'humitat.

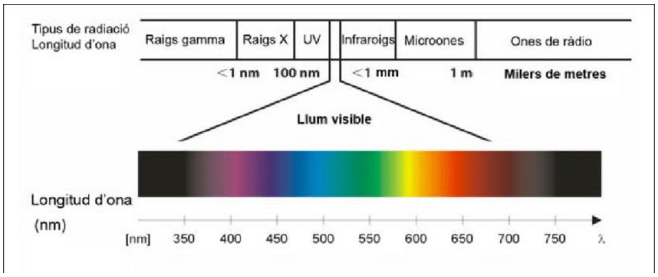


Il·lustració 28: Sensor d'humitat

- Magnètics

Els imants o els corrents elèctrics generen camps magnètic, doncs aquests sensors, tenen la capacitat de detectar-los. Els principals en són els interruptors Reed. Aquests consten de dues làmines metàl·liques fetes amb materials ferromagnètics introduïdes en una càpsula que quan estan en un camp magnètic s'atrauen provocant un tancament del circuit.

- D'infrarojos



Il·lustració 29: Sensor d'infrarojos

En l'espectre electromagnètic, on s'hi recullen tots els tipus d'ones juntament amb la seva freqüència (f) i longitud d'ona (λ), es troben unes amb una freqüència massa baixa com per poder ser detectades per l'ull humà, aquestes són els infrarojos.

Hi ha uns dispositius semiconductors anomenats díodes, uns interruptors que permeten que passi el corrent en una sola direcció. Són també coneguts com a "rectificadors", ja que canvien de CA (corrent altern) a CC (corrent continu). Hi ha uns díodes que poden emetre llum infraroja i uns transistors sensibles a aquestes ones que detecten les emissions que fan els díodes. Així és, per exemple, el funcionament dels comandaments a distància, aquests tenen díodes que emeten infrarojos a través dels fototransistors del mateix.

COM ES CREA UN PROGRAMA INFORMÀTIC?

Per a crear un programa informàtic, el procés es divideix en 3 fases; l'analítica, on decideixes quines funcions vols que realitzi el programa; la programació, on, a partir d'un llenguatge de programació, es codifiquen les funcions pensades anteriorment, i la revisió, on es comprova que el programa funcioni correctament i d'acord amb les característiques i expectatives emprades en la primera fase, que ha servit per optimitzar el temps de desenvolupament d'aquest.

QUÈ ÉS UN COMPILADOR?

És un Software que converteix el programa escrit a un llenguatge de programació d'alt nivell que pugui ser interpretat per una màquina. Aquests utilitzen algoritmes, és a dir, instruccions de l'execució d'un problema, concretant l'objectiu i el final, detallant els passos per resoldre-ho. Després, és quan es tradueixen al llenguatge que l'ordinador sigui capaç de descodificar. A més, aquests llenguatges de programació d'alt nivell, són comprensibles per a totes les persones, ja siguin expertes o no. Com són així de senzills, dins la seva complexitat darrere les pantalles, tenen una gran qualitat de manipulació i execució dels programes creats en aquests. Antigament, no s'havia avançat tant en informàtica i només hi havia els llenguatges per programació de baix o mitjà nivell, els quals no eren així d'eficaços.

Un compilador és una de les bases de la programació, i és molt útil per a entendre com funciona la comunicació entre un llenguatge màquina d'alt nivell i un ordinador compost per xips i transistors. Poder conèixer el funcionament d'això, permet desenvolupar i programar d'una manera més precisa els llenguatges d'alt nivell.

ELS LLENGUATGES DE PROGRAMACIÓ

Els llenguatges de programació són un sistema de comunicació amb estructura, ús i contingut determinats que passa unes instruccions de tal manera que un ordinador pugui produir certes dades i accions. Per a fer-ho es basen en la codificació de la informació i la transformació d'un llenguatge que nosaltres entenem, el natural, a un amb dades numèriques que l'ordinador pugui interpretar. Així doncs, permeten la comunicació amb la màquina per dur a terme certes accions requerides.

Es poden fer servir per crear programes que tenen un control sobre el comportament de la màquina o per mostrar algoritmes precisos. Tenen el nom de "llenguatge" perquè estàn compostos per símbols que determinen l'estructura i el significat d'elements i expressions.

Hi ha diferents maneres de classificar els llenguatges de programació, segons el nivell d'abstracció (alt i baix nivell), segons la forma d'execució i de processar ordres, segons el paradigma de programació que utilitzen, etc. D'aquesta última els més importants en que es classifiquen són; el paradigma imperatiu i el funcional/estructurat. Tot i això, també hi ha altres com el paradigma orientat a objectes o el lògic.

- Paradigma imperatiu

Aquest tracta d'una sèrie d'instruccions o ordres que influeixen en l'estat d'un programa. Té unes ordres condicionals que deixen al programa continuar un bloc d'ordres si es dóna algunes de les condicions. Així van ser els primers llenguatges de programació i és el paradigma en que es basa la màquina (0 i 1). Avui en dia encara hi ha llenguatges que utilitzen aquests principis. Dins d'aquest paradigma hi trobem el paradigma procedimental que és el contrari del paradigma declaratiu, i una mancança que presenta el paradigma imperatiu és la falta de flexibilitat a causa de les seqüències d'instruccions.

- Paradigma funcional/estructurat

La programació es divideix en blocs (procediments i funcions) que es poden comunicar entre si o no. Cada funció recull un procediment tancat i es podria

entendre com un petit programa en si mateix. Les funcions permeten, mitjançant la subdivisió dels processos en petites accions, l'aprofitament del codi per a altres objectius o en altres parts d'un programa.

Els llenguatges també es classifiquen entre els llenguatges interpretats i els compilats:

- Llenguatges interpretats

Com ja sabem, la màquina només comprèn el codi binari, així doncs, els llenguatges interpretats són els que no estàn en codi binari i requereixen d'un programa extern, que s'anomena intèrpret, que tradueix el llenguatge per tal que la maquina sigui capaç de processar-lo i executar-lo. Alguns exemples en són el BASIC, el MATLAB, el PHP i el Perl.

- Llenguatges compilats

En aquests, la transformació a un llenguatge que la màquina pugui entendre abans d'acabar el programa, la fa un programa adjunt que s'anomena compilador. L'arxiu que surt s'anomena arxiu executable, i com el seu nom diu, es pot executar sense requerir de cap altre programa o suport extern, tot i això, té un punt en contra ja que si s'ha de fer alguna modificació en el codi, o en les fonts del programa, obligatoriament s'ha de fer una recompilació del programa per a poder aplicar els canvis, i això els fa menys flexible. Alguns exemples en són el C, el C++, el COBOL, l'ADA i el Pascal.

Una altra manera molt important de classificar els llenguatges de programació són en els de baix nivell i els d'alt nivell.

- Llenguatges de programació de baix nivell

Són aquells que no poden ser entesos per totes les persones i un exemple n'és el codi binari.

- Llenguatges de programació d'alt nivell

Aquests tenen en compte les capacitats de qualsevol persona per tal que l'usuari pugui resoldre problemes senzilla i ràpidament. Alguns exemples en són el PHP, el Phyton, el Perl i el Java.

LLENGUATGES DE PROGRAMACIÓ MÉS POPULARS

Hi ha molts llenguatges de programació però a continuació anomenaré els més importants, si més no, els més destacables:

- PHP

Aquest llenguatge, com l'HTML, que explicaré més tard, i juntament amb aquest té una utilitat principal que és fer-lo servir per a les pàgines web. Permet crear pàgines amb contingut dinàmic o enviar i rebre *cookies*, entre altres. Es basa en "*scripts*", que és un guió d'ordres que rep un servidor d'una pàgina web per a poder llegir el codi font, en el servidor. Va ser dels primers que es podien introduir en un document HTML. Per al seu funcionament, un cop introduït un codi, el servidor l'interpreta i genera la pàgina web.

Il·lustració 30: PHP

- Phyton

Guido Van Rossum l'informàtic que als 80 va desenvolupar aquest llenguatge, ho va fer amb la intenció de que fos molt senzill d'aprendre i fer servir. Les seves comandes són paraules angleses, així que el seu codi és fàcil d'interpretar, ja que paraules com *"print"* per imprimir un text o "*if"/"else"* com a condicions fan que sigui realment senzill de llegir. Per tant, converteix aquest llenguatge en un molt útil per desenvolupar aplicacions sense problemes, i fins i tot permet trobar errors tot i que el programa l'hagi escrit una altra persona.

Aquest llenguatge és el més utilitzat en finances, també és molt útil per a realitzar càlculs o aplicar fórmules. A més, Google o la NASA, per exemple, també l'utilitzen en els seus projectes. També és el llenguatge en el qual es basen l'Instagram, el Pinterest o el YouTube. Tot i això, té dos punts en contra que són que funciona bastant lent a l'hora d'executar les tasques i que no és gaire encertat per a crear apps pels mòbils o aplicacions web, que són aquelles que estan instal·lades en un servidor i per executar-les es requereix d'un dispositiu amb connexió a Internet i un navegador, com per exemple Google Chrome o Mozilla Firefox.

Il·lustració 31: Instagram

- Java

Aquest llenguatge és el més utilitzat en tot el món segons l'índex TIOBE, tot i que a vegades l'ha atrapat el llenguatge C, que comentarem més tard. L'any 1995, "Sun Microsystems" va desenvolupar aquest llenguatge tan versàtil, és a dir, útil per a moltes funcions, per no dir gairebé tot. Per exemple per crear jocs com el "Minecraft", apps per a Android, programes d'ordinador, aplicacions web o aplicacions de Google o Ebay, per exemple. Una característica que té és que és independent del hardware, és a dir, pot funcionar en qualsevol mòbil o ordinador mitjançant un intèrpret de Java, i això produeix l'anomenat *"WORA"* *(*"write once, run anywhere") que bàsicament vol dir que es pot executar a qualsevol lloc un cop compilat. És un llenguatge orientat a objectes, que vol dir que hi ha uns determinats blocs de programació que accepten certes dades i, a partir d'això, donen certs resultats. Els programadors creen aquests objectes i els comparteixen de manera que tothom els pugui utilitzar. Així doncs, com moltes funcions i eines ja són existents, només cal afegir-les a les aplicacions que l'usuari estigui manipulant. També és un derivat de C i C++ i s'ha de compilar.

Il·lustració 32: Java

- Javascript

Aquest altre llenguatge també molt conegut i senzill, no s'ha de confondre amb el Java, tot i que contingui elements d'aquest i del llenguatge C, ja que no s'utilitzen igual. S'utilitza en pàgines web per donar-li petites o grans animacions, textos, sons, etc. Com bé diu el seu nom, es basa en *"scripts"*, anomenats anteriorment. El Javascript, és el més utilitzat en aplicacions dins de pàgines web, ja que funciona directament al navegador. Així doncs, per fer-lo anar només s'ha d'escriure els programa en format de document de text i seguidament obrir-lo al navegador a mode de pàgina web, i aquest el farà funcionar. Aquest llenguatge també està orientat a objectes. Com la majoria, aquest també té un punt negatiu i és que com he dit anteriorment, només serveix per crear eines i funcions dins una pàgina web, així doncs, per utilitzar-

lo també s'han de conèixer altres llenguatges que serveixen per crear les pàgines, com per exemple l'HTML o el PHP.

- Go

Aquest llenguatge va ser desenvolupat per Google l'any 2009. Té com a objectiu diferents elements d'altres llenguatges com el C, el Phyton o el C++. És un llenguatge per procediments, que vol dir que el programa es divideix en uns certs components o procediments, i aquests es poden utilitzar en diferents parts del programa. Per exemple, un programa que serveix per a dividir es pot fer mitjançant l'ús del component "resta", que es repeteix moltes vegades per tal de dur a terme la divisió. És un llenguatge senzill però alhora serveix per crear tota mena d'aplicacions web. Per exemple Netflix i Dropbox estan, en part, desenvolupades amb Go.

Il·lustració 33: Netflix i Dropbox

- Kotlin

Aquest llenguatge va ser desenvolupat a Russia el 2016. Tot i que no sigui tan popular com altres, és molt utilitzat en el desenvolupament d'aplicacions per Android, en el desenvolupament web, en aplicacions científiques i aplicacions multiplataforma funcionals tant per iOS com per Android. El llenguatge Kotlin pretén substituir el llenguatge Java, oferint les mateixes capacitats, però amb un funcionament més senzill.

- Scala

Scala és un llenguatge de programació desenvolupat per Martin Odersky l'any 2001. Aquest s'executa en la màquina virtual de Java, que és un software que té la capacitat de carregar dins seu un altre sistema operatiu per tal de fer

veure que és un ordinador de debò, per tant, bàsicament és una màquina, que en comptes de ser física, és virtual, una simulació. També pot utilitzar llibreries de Java i Javascript fent-lo així més fàcil d'introduir en actualitzacions d'aplicacions preexistents. Twitter o BBVA, en són exemples d'empreses que han utilitzat Scala en algunes de les seves aplicacions. És un llenguatge funcional, que vol dir que es basa en funcions que es poden executar en diversos nuclis de processador o diversos servidors en una plataforma que es troba a la "nube" o el núvol. Això fa que s'utilitzi molt en aplicacions d'anàlisis de dades i serveis al núvol.

- Ruby

Aquest llenguatge el va desenvolupar l'informàtic japonès Yukihiro Matsumoto. És molt adequat pels principiants, ja que el seu objectiu és ser divertit d'aprendre i utilitzar i fer que la màquina s'adapti a les persones en comptes de ser al contrari. És utilitzat per a crear aplicacions web, però també serveix per a crear apps pel mòbil, o aplicacions d'escriptori que són aquelles que estan instal·lades a l'ordinador de l'usuari i s'executa a partir del sistema operatiu com per exemple Windows, Linux o macOS. Ruby és un llenguatge orientat a objectes.

- Swift

Aquest llenguatge va ser desenvolupat per Apple, i l'oficial d'aquest, per tal de crear apps per iOS, Mac, Apple Watch, etc. El seu ús és ràpid i senzill. Segons Apple, és 8'4 vegades més ràpid que Phyton. LinkedIn, per exemple seria un exemple d'aplicació per iOS que ha estat desenvolupada amb Swift. Tot i que pertany a Apple és gratuït i de codi obert. A més, és un altre llenguatge orientat a objectes.

Il·lustració 34: Linkedin

- COBOL

El COBOL és un llenguatge de programació d'alt nivell que va ser dissenyat per al desenvolupament dels negocis, i està orientat a arxius i aplicacions. Aquest no pot ser utilitzat per escriure programes de sistemes com el sistema operatiu

o el compilador. Ha sigut el més utilitzat durant molts anys, però actualment i cada cop més, hi ha altres llenguatges com Java o Phyton que s'estan posant a la seva alçada i, fins i tot, estan superant-lo.

- C++

Es tracta d'un llenguatge que buscava ser una extensió d'un altre llenguatge anomenat C, per tal de tenir la capacitat de manipular objectes, per tant, és orientat a objectes. El danès Bjarne Stroustrup el va desenvolupar als anys 80. Aquest llenguatge ofereix un alt rendiment però per altra banda té la seva part dolenta. El C++ és un llenguatge amb un codi molt ampli, a més, necessita una compilació per cada plataforma, i dominar les llibreries és més complicat que amb altres llenguatges anomenats anteriorment. Tot i això és molt usat, com per exemple en navegadors web, ja que requereixen d'aquesta rapidesa que els proporciona, o també en aparells mèdics, per programar els gràfics dels videojocs o per fer rellotges intel·ligents.

- C#

Aquest llenguatge pronunciat com a "*C Sharp*", va ser creat per Microsoft i també està orientat a objectes. És també una extensió del C i del C++. Microsoft el va dissenyar per tal que es poguessin crear aplicacions a la seva plataforma (*.NET*) de manera senzilla i intuïtiva. Aquest llenguatge permet que totes les dades obtingudes programant-lo es quedin registrades i guardades per a la seva posterior utilització. El C# serveix per a crear videojocs, per a desenvolupar el software per sistemes operatius de Windows…

- ASP

Active **S**erver **P**ages és un llenguatge de programació desenvolupat per Microsoft l'any 1998. S'executa en el servidor web abans de ser enviat a la pàgina web a través de l'Internet que posseeix l'usuari. Les pàgines executades realitzen diverses tasques per a crear la pàgina resultant que es mostra a l'usuari, el qual obté la pàgina web en codi HTML, fent-lo així compatible amb qualsevol navegador, fruit de l'execució de la pàgina ASP. Més recentment s'ha

realitzat una actualització de l'ASP anomenat ASP.NET, el qual té una manera d'utilitzar-se diferent.

- MYSQL

Basat en el llenguatge SQL, MySQL és un sistema de gestió de bases de dades. Aquest es pot executar en les plataformes Windows, MacOS i Linux. Té bastantes utilitats però entre les més destacades s'hi troben gestionar aplicacions web, publicar contingut en línia, etc. A més, és utilitzat al WAMP (per a Windows) i al LAMP, una plataforma de desenvolupament web que té Linux com a sistema operatiu, Apache com a servidor web, PHP com a llenguatge de programació, tot i que a vegades s'usa Perl o Phyton, i MySQL com a sistema de gestió.

- HTML

HTML que significa *"HyperText Markup Language"* traduït com a "llenguatge de marques d'hipertext", és un llenguatge que serveix per a definir el text i elements diversos que es mostren a les pàgines web que ens trobem diàriament a Internet. Aquest llenguatge s'utilitza, majoritàriament, per crear pàgines web i és relativament senzill d'aprendre.

- CSS

CSS que significa "*Cascading Style Sheets*" traduït ordinàriament com a "fulles que serveixen per donar estil a les pàgines". Això ho fa donant el color, definint el fons, les tipografies (tipus de lletra), etc. Així doncs, la seva funció és bàsicament donar-li una forma atractiva a la pàgina web anteriorment creada en estructura, potser, amb HTML.

LLENGUATGES QUE UTILITZARÉ

Per desenvolupar la part pràctica del meu treball utilitzaré tres llenguatges de programació; HTML, CSS i JavaScript. He triat aquests perquè són més senzills d'aprendre en poc temps que altres. També perquè tenen un entorn gràfic fàcil de programar, i a més no es requereix de cap compilador per poder executar-los. Així doncs es pot fer en qualsevol navegador. En cas de que hagués triat uns altres diferents necessitaria compiladors i a l'hora de presentar-ho al treball seria més complicat, i necessitaria bastant més temps que els escollits. Tot i això, aprendre HTML, CSS i JavaScript, per algú que no ha programat mai, porta al voltant de 2 anys, així que he hagut de sintetitzar al màxim els continguts per poder fer-ho en tan sols uns mesos.

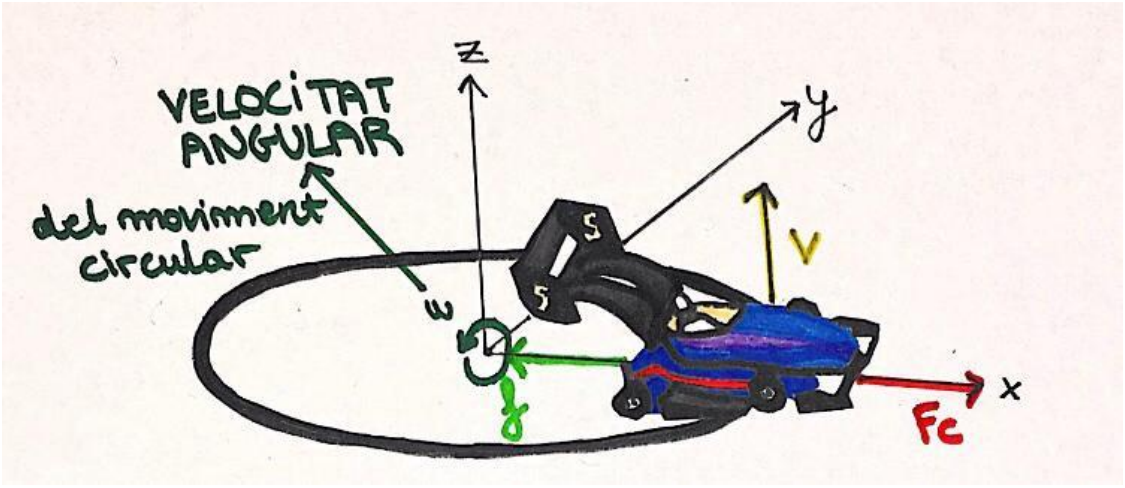
He pensat en utilitzar els tres i no un perquè es complementen, no podria arribar a realitzar el meu treball amb només un d'ells. HTML serveix per definir el contingut de la pàgina i prou, CSS a partir de l'"*Style"*, serveix per proporcionar el disseny desitjat d'aquesta, i JavaScript és l'encarregat, a partir de l'*"Script*", de programar el comportament de la pàgina, és a dir, definir les funcions que es duran a terme a partir dels objectes que proporcionen els altres dos.

Il·lustració 35: Llenguatges que utilitzaré

TREBALL PRÀCTIC

PART FÍSICA

Quan vaig iniciar el meu treball tenia clar que volia fer una simulació. Primer vaig buscar informació, la qual surt explicada en la introducció i també en el marc teòric. Una vegada ja estava més endinsada en el tema, se'm va ocórrer que podria fer una simulació de cotxes on es veiés què succeeix quan un cotxe va massa ràpid en una corba. Per a fer-ho utilitzaré els llenguatges HTML, CSS i Javascript pels motius anomenats en l'apartat anterior. Necessitaré saber els paràmetres que serviran determinar l'equació del moviment del cotxe, i també les lleis del sistema al qual pertanyen. Això ho aconsegueixo partint de les lleis de la dinàmica que he estudiat a física, més concretament la Segona Llei de Newton, també anomenada Llei Fonamental; (suma de forces és igual a massa per acceleració). Junt amb el diagrama de forces de sota, surt l'equació que busco partint de la llei:



$$\sum F = ma$$

$$\sum F = Força\ de\ fregament - força\ centrípeta$$

On la suma de les forces està composta per les dues forces que observem al diagrama. L'assenyalada de color verd, és la força de fregament ($\mu \cdot m \cdot g$) que buscarà oposar-se a la força assenyalada en vermell, la força centrípeta o acceleració centrípeta ($m \cdot \frac{v^2}{R}$), per poder mantenir el cotxe a la carretera.

Així doncs, podem veure que els paràmetres que intervenen són; el fregament, compost pel coeficient de fregament (μ), multiplicat per la força normal (N) que és la que actua "en contra" del pes, fent així que el terra no s'enfonsi, i que es calcula igual que el pes (m·g), i l'acceleració centrípeta, composta per la "m", i la velocitat al quadrat dividida entre el radi, ja que aquests són els que serveixen per calcular el moviment circular, important en el meu treball, ja que el cotxe que simularé donarà voltes a una rotonda. Llavors tenim que:

$$\sum F = \mu \cdot m \cdot g - m \cdot \frac{v^2}{R} = m \cdot a$$

D'aquí podem simplificar les masses i llavors quedarà de la següent manera:

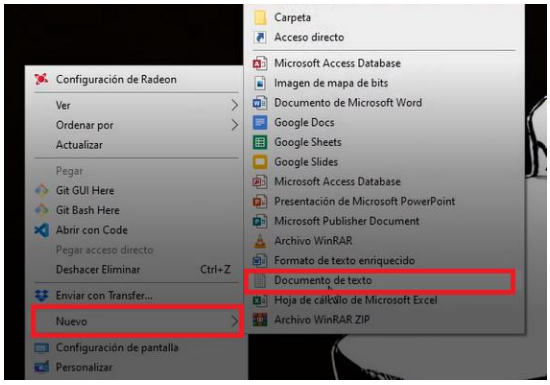
$$\mu \cdot g - \frac{v^2}{R} = a$$

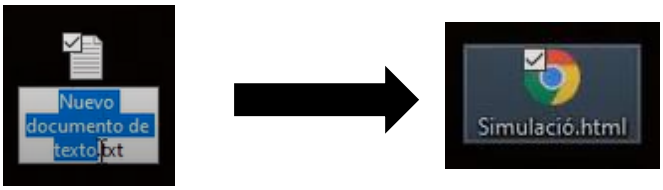
Un cop obtinguda aquesta expressió, si el número obtingut de $\mu \cdot g$ és més gran que l'obtingut de , no hi haurà moviment, ja que la força de fregament no pot crear el moviment de l'acceleració, així doncs l'a serà 0. En canvi, si el número obtingut de $\frac{v^2}{R}$ és més petit que l'obtingut de , el cotxe experimentarà una acceleració que buscarà fer-lo fora de la corba. Si això passa serà perquè haurà agafat una velocitat tan elevada que la força de fregament no tindrà prou força per a retenir-lo.

Llavors, sabem que si l'acceleració és positiva, la força de fregament "guanya" i manté el cotxe en el seu camí, en canvi, si l'acceleració és negativa, la força de fregament "perd" i el cotxe surt de la carretera.

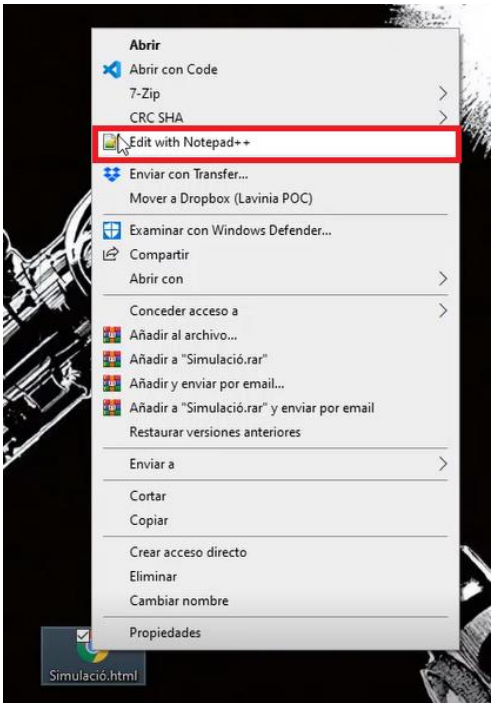
PROGRAMACIÓ

Un cop tinc això, començo a programar els tres llenguatges de programació que eren completament desconeguts per mi. El primer que hauré de fer és començar amb la base. A l'escriptori mateix, fent "click" al botó dret trobo una opció de crear un arxiu nou, i trio un document de text, un cop fet això canvio l'extensió del nom de l'arxiu de ".txt" a ".html". Fet que converteix l'arxiu a una pàgina web.

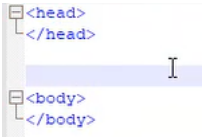




Un cop fet això, obro l'arxiu amb una aplicació anomenada "NotePad++", que serveix per programar, però ho podria fer al bloc de notes mateix.



Utilitzant el codi HTML, aniré afegint els objectes que componen el simulador. Els que divideixen l'estructura són el «<head>» i el «>body>» , definits amb etiquetes d'obertura i de tancament. (Si cal, pots consultar l'annex).



Però dins d'aquests també hi haurà d'altres com imatges, àudios o botons.

Creo els títols que es veuran quan obrim la pàgina a un navegador, i el títol que es veurà quan passem el ratolí per la pestanya.

A partir d'aquí afegeixo parts dins el head. Primer introdueixo l"<style>", construït amb CSS, el qual s'encarrega de la part visual de la pàgina, és a dir, són com els elements que es veuen en aquesta.

Després introdueixo l"<script>" que com bé diu el seu nom és el guió, o més aviat el guionista, ja que és el que s'encarrega de dur a terme les funcions de la pàgina, és a dir, totes les accions, i aquest es construeix amb el JavaScript, però que deixaré per l'últim, ja que és el més complicat i també perquè primer he de construir els elements que han de dur a terme les accions.

Aquests elements els començo a crear a la següent estructura, el "<body>". Introdueixo com «caixes» on hi estan continguts els elements que vull crear ja siguin elements individuals o grups d'elements.

```
<body>

<div id="centrar">
<div id="pista">
<div id="coor">
 <img id="cotxe" src="img/cotxe.png">
</div>
</div>
</div>

</body>
```

Dins d'aquestes caixes creo uns objectes que contenen quadres de texts i botons, als quals donaré vida en l'"<script>" més endavant.

```
<div>
 <hr>

 <input type="text" id="velocitat" placeholder="Introduir velocitat..."></input><br>
 <br>
 <input type="text" id="fregament" placeholder="Introduir coeficient de fregament..."></input><br>
 <br>
 <input type="button" id="boto" value="Click" onclick="correr();"></input>
 <input type="button" id="buto" value="Reset" onclick="window.location.reload(true)"></input>
</div>
```

Cercant informació de manera exhaustiva trobo els paràmetres amb els quals funciona cada element, que són molt extensos i per tant no explicaré de manera detallada.

Aquests paràmetres els introdueixo per modificar amb CSS l'"style", i cada element l'anomeno pel seu identificador.

Construeixo unes coordenades amb unes certes mesures per a saber on col·locar la pista i el seu respectiu cotxe.

```
#coor{
position:absolute;
top:0px;
left:0px;
width:150px;
height:150px;
}
```

Com prèviament sé que utilitzaré el radi, com hem vist en el plantejament de les equacions, el determino amb 150px (píxels), perquè es pugui veure bé a totes les pantalles. Totes les mesures següents es regiran per aquesta. Un cop tinc la pista col·locada vaig afegint detalls perquè quedi més vistosa.

Tot els elements agafaran posicions uns respecte els altres amb el paràmetre «position:absolute».

Un cop tinc la pista, col·loco el cotxe mitjançant unes coordenades.

Al cotxe, l'hi afegiré un altre paràmetre anomenat "z-index" que servirà per posar capes a les caixes, fent així que aquest ocupi la posició més avançada, i per tant, es veurà davant de tot. El dimensiono perquè tingui unes mides on es vegi bé. I ara centro tots aquests elements.

Quan tinc això fet, decoro el fons amb una foto, que buscant com posar els paràmetres que vull, aconsegueixo difuminar-lo.

Després afegeixo dues imatges que seran uns logos que he triat i els hi introdueixo unes característiques de la meva elecció.

Un cop feta aquesta part, començo a donar vida als elements mitjançant l'"script". Aquest apartat l'explicaré més detalladament:

```
t=0;
```

1. Aquesta instrucció fa que quan la pàgina es carrega el temps val 0.

```
R = 150;
```

2. Aquesta indica que el radi mesura 150.

```
function simular(){
```

3. Aquesta diu que s'introdueix una funció anomenada "simular".

```
T=t/10;
```

4. Aquesta instrucció fa que el temps es faci 10 vegades menor amb l'objectiu de que el cotxe es vegi movent-se de la manera correcta, ja que si no com cada segon que passa la posició és una determinada, el que es veuria seria la imatge del cotxe a aquestes i no es mouria seguidament. Per això, en dividir-lo entre 10, s'omplen més espais i es veu com si realment es mogués lliscant.

```
v = document.getElementById("velocitat").value;
```

5. Aquesta diu que la velocitat es determina amb aquest paràmetre, on la paraula entre cometes és un objecte creat al "<body>".

```
f = document.getElementById("fregament").value;
```

6. Aquesta fa el mateix que l'anterior però amb el fregament.

```
//alert("Velocitat: "+velocitat);
//alert("Fregament: "+fregament);
```

7. Són "*alerts*" que serveixen per comprovar que el que vaig fent es carrega de la manera correcta.

```
w = v/R;
```

8. És la fórmula que serveix per calcular la velocitat angular.

```
a = f*9.8-(v**2/R);
```

9. És la fórmula desenvolupada que serveix per calcular l'acceleració.

```
x = R+180*Math.cos(w*T);
y = 180*Math.sin(w*T);
```

10. Són les fórmules desenvolupades que serveixen per calcular les posicions en x i en y, que són les que es mostren en dues dimensions.

```
if(a>0){
```

11. És un condicional que diu que si l'acceleració és més gran que 0 passarà alguna cosa.

```
document.getElementById("cotxe").style.left = x+"px";
document.getElementById("cotxe").style.bottom = y+"px";
```

12. Si es dona el cas de la instrucció anterior, aquesta col·locarà el cotxe en l'eix x i en l'eix y dins la pista on li pertoqui.

```
}else{
```

13. És l'altra part del condicional que indica què passarà si l'acceleració no és més gran que 0.

```
R = 150+0.5*a*T**2;
```

14. Indica que el radi serà més gran que de costum, i per tant, el cotxe se sortirà de la pista, i aquesta instrucció va acompanyada de:

```
document.getElementById("cotxe").style.left = x+"px";
document.getElementById("cotxe").style.bottom = y+"px";
```

```
}
```

15. Indica que el condicional creat ha acabat.

```
if(y>200 || x>350 || x<-150 || y<-300){
```

16. És un altre condicional que diu que si aquells paràmetres es compleixen, voldrà dir que el cotxe ha sortit de la pista.

```
clearInterval(grijo);
```

17. Serveix per aturar la funció, i “grijo” és una variable que podria tenir qualsevol nom, però cada vegada que es nomena, ha de tenir-ne mateix.

```
t=0;
```

18. Com s’atura la funció, aquesta instrucció indica que el temps ha de tornar a ser 0.

```
document.getElementById("cotxe").src = "img/pum.gif";
```

19. Diu que introduint aquests paràmetres, quan el cotxe se surt, sortirà un *gif* que he triat.

```
}
```

20. Indica que el condicional creat ha acabat.

```
t++;
```

21. Indica que sigui quin sigui el valor de t, sempre li sumarà 1.

```
if(t>5000){clearInterval(grijo);}
```

22. És un altre condicional que indica que si arriba un punt on el temps transcorregut a la simulació és major a 5000, la simulació s’atura.

```
}
```

23. Indica que la funció creada ha acabat.

```
diode = 0;
```

24. Indica que díode, una altra variable que indica si la funció està en marxa o no, val 0, és a dir, no està en marxa.

```
function correr(){
```

25. Aquesta instrucció diu que s'introdueix una funció anomenada "correr".

```
if(diode == 1){
```

26. És un altre condicional que indica que si díode en comptes de ser 0 val 1, és a dir, està en marxa, passarà el següent:

```
clearInterval(grijo);
```

27. Instrucció que farà aturar la funció, i per tant la simulació.

```
diode = 0;
```

28. Indica que díode torna a valer 0.

```
t=0;
```

29. Indica que el temps també serà 0.

```
}else{
```

30. És l'altra part del condicional que indica què passarà si díode no val 1.

```
grijo = setInterval("simular()",25);
```

31. Ordena que s'iniciï la funció "simular".

```
diode = 1;
```

32. Indica que en aquell moment díode tornarà a valer 1.

```
}
```

33. Indica que el condicional creat ha acabat.

```
}
```

34. Indica que la funció creada ha acabat.

CONCLUSIONS

A l'inici, en la meva ment només hi havia la idea d'automatitzar les operacions per tal de que no fos tan difícil per les persones calcular. Per a fer-ho, vaig pensar en les simulacions i després em va sorgir la idea de que podria programar la meva pròpia simulació. Sabia que era un camí llarg i difícil, però havia d'intentar-ho.

El meu objectiu era bàsicament aquest, aconseguir fer una simulació d'algun problema de física com els que fem a classe. Un cop feta, puc dir que ho he aconseguit amb escreix.

Tot i això i com era d'esperar, hi ha matisos a perfeccionar dins de la meva simulació, però per haver començat des de 0 en aquest grandíssim món de la informàtica, he arribat a construir una simulació que mai hagués esperat fer. Ha sigut bastant complicat tot plegat, ja que abans de la part pràctica he hagut de documentar-me molt sobre la història que ens ha portat on estem avui. Tots els avenços tecnològics dels quals gaudim en el nostre dia a dia ha sigut fruit de petites o grans aportacions de diferents persones al llarg d'uns 21 segles.

Centrant-nos més específicament de la meva simulació, nomenaré alguns factors que no han resultat com m'hagués agradat:

- Hi ha vegades que quan el cotxe se surt de la pista, depenent de la velocitat i el fregament atorgats, surt d'una manera que no hauria de sortir, fent una volta innecessària, que no sorgiria a la vida real. Això passa perquè tot i ser una simulació molt avançada pel meu nivell, no està composta per paràmetres més complicats que es fixen en petits detalls. Independentment d'aquest petit detall, estic molt satisfeta amb la feina feta, ja que en general, com he dit anteriorment, els estudiants de programació triguen bastant més temps del que jo he tingut en desenvolupar-se amb soltesa.
- La pista és una rotonda perquè una el·lipse requeria d'uns paràmetres i unes equacions fora del meu coneixement assolit aquests anys. De totes maneres, una rotonda és més que realista i vistosa.

- El so de la pista d'àudio que he inclòs dins la simulació torna a iniciar-se quan li donem al botó *reset*, ja que aquest botó s'encarrega de carregar la pàgina sencera de nou. Si hagués tingut més coneixements, suposo que podria haver buscat la manera de que no s'aturés independentment de la resta de la simulació.
- Els paràmetres per modificar els botons i posar-los més bonics, eren limitats, i per tant, no estan tan ben fets com hagués volgut.

En conclusió, estic molt contenta amb els resultats perquè és com ho imaginava i també bastant divertit. No obstant això, també he pogut observar que per crear una simulació, no només hi ha prou amb saber programar, sinó que també s'ha de tenir un cert coneixement sobre el camp que es vol simular. Com per exemple, en el meu cas, sense els meus coneixements sobre física no hagués pogut saber quines equacions necessitava per fer que el cotxe funcionés. He invertit moltes hores, i al final he obtingut una simulació feta per mi que automatitza les operacions i llavors només s'han d'aportar unes dades i es realitza què passaria a la vida real.

BIBLIOGRAFIA:

ACEVEDO-DÍAZ, José Antonio. *Sobre modelos científicos* [en línia]. Huelva: editorial, 2017, 21 de maig. Disponible a: <https://www.oei.es/historico/divulgacioncientifica/?Sobre-modelos-cientificos#_ftn1>. [Consulta: 14 febrer 2020].

ACEVEDO-DÍAZ, José Antonio; GARCÍA-CARMONA, Antonio; ARAGÓN, María del Mar; OLIVA, José María. *Modelos científicos: significado y papel en la práctica científica* [en línia]. Bogotà, Colòmbia: Revista Científica, 2017, setembre. Disponible a: <https://www.researchgate.net/publication/319457577_Modelos_cientificos_significado_y_papel_en_la_practica_cientifica>. [Consulta: 25 febrer 2020].

ADELL, Ferran. *Llenguatges de programació: classificació, tipus i recursos d'aprenentatge* [en línia]. Catalunya, Espanya. Disponible a: <http://multimedia.uoc.edu/blogs/fem/lenguajes-de-programacion-clasificacion-tipos-y-recursos-de-aprendizaje/>. [Consulta: 20 juliol 2020].

ALTAMIRANO, Eduardo. *linea del tiempo de la simulación* [en línia]. Espanya: 2016, 01 de febrer. Disponible a: <https://prezi.com/dupvtbwn3vl3/linea-del-tiempo-de-la-simulacion/>. [Consulta: 01 maig 2020].

ALTAIR. Túnel de viento virtual de hyperworks de altair para una simulación aerodinámica más rápida y exacta [en línia]. Espanya: 2013, 09 d'octubre. Disponible a: <https://www.3dcadportal.com/cad-news/2455-tunel-de-viento-virtual-de-hyperworks-de-altair-para-una-simulacion-aerodinamica-mas-rapida-y-exacta.html>. [Consulta: 19 febrer 2020]

ALVAREZ, Miguel Angel. *Qué es ASP* [en línia]. València, Espanya: 2001, 09 de maig. Disponible a: <https://desarrolloweb.com/articulos/393.php>. [Consulta: 25 juliol 2020].

Ana Lucia. *PIONEROS DE LA COMPUTACIÓN* [en línia]. Londres, Anglaterra: 2020. Disponible a: <https://www.timetoast.com/timelines/pioneros-de-la-computacion--22>. [Consulta: 04 maig 2020].

ANÒNIM. *Tipos de sensores* [en línia]. 2013, 29 de maig. Disponible a: <http://rincon-escondido-lainformatica-luisa.blogspot.com/2013/05/sensores-el-sensor-la-informacion-que.html>. [Consulta: 17 maig 2020].

ANFO. *Aplicaciones de Escritorio y Web* [en línia]. 2013, 20 d'agost. Disponible a: <*https://es.slideshare.net/anfo24/aplicacion-escritorio-web-25427914*>. [Consulta: 17 agost 2020].ASTESANA, Sebastian. *La máquina de von Newman. El IAS* [en línia]. Argentina: 2012, 3 de gener. Disponible a: <https://www.taringa.net/+ciencia_educacion/la-maquina-de-von-newman-el-ias_v1aqw>. [Consulta: 25 febrer 2020].

AYS. *SIMCENTER 3D* [en línia]. Espanya: 2020. Disponible a: <http://www.aysplm.com/portfolio-item/simcenter-3d/?gclid=Cj0KCQjwiYL3BRDVARIsAF9E4Gf5Jpy2shAUdkSmRYA9QvW17r4IIA91Gg8BhXxqxNBUfzjOeQpZyzkaAlleEALw_wcB#casosdeexito>. [Consulta: 25 maig 2020].

BANCO SANTANDER. *Simulador de hipoteca* [en línia]. Espanya: 2020, 25 de febrer. Disponible a: <https://www.bancosantander.es/es/particulares/hipotecas/simulador>. [Consulta: 25 febrer 2020].

BANKIA. Simulador de hipotecas [en línia]. Espanya: Bankia, S.A., 2020. Disponible a: <https://www.bankia.es/es/particulares/financiacion/hipotecas/simulador-hipotecas>. [Consulta: 25 febrer 2020].

BBC NEWS MUNDO. *Al-Juarismi, el erudito persa que introdujo los números a Occidente y nos salvó de tener que multiplicar CXXIII por XI* [en línia]. 2018, 05 d'agost. Disponible a: <https://www.bbc.com/mundo/noticias-44933192>. [Consulta: 17 maig 2020].

BBVA. *Simulador de hipotecas* [en línia]. Espanya. Disponible a: <https://www.bbva.es/personas/productos/hipotecas/simulador-hipotecas.html>. [Consulta: 25 febrer 2020].

BELLED, Sara. *Simulador interactivo: Cómo controlar la curva del coronavirus en plena desescalada* [en línia]. Astúries, Espanya: 2020, 11 de maig. Disponible a: <https://www.elcomercio.es/sociedad/salud/coronavirus-graficos/simulador-interactivo-controlar-20200511105058-ntrc.html?ref=https:%2F%2Fwww.google.com%2F>. [Consulta: 22 maig 2020].

BOLINCHES, Vicente. *Tipo ME2X - Unidad de control del sistema* [en línia]. Barcelona, Espanya: Bürkert Ibérica S.A.U., 2020. Disponible a: <https://www.burkert.es/es/type/ME2X>. [Consulta: 27 juny 2020].

BUBENKO, J.A.. *SIMULATING A "MANAGEMENT GAME" WITH PROGRAMMED DECISIONS* [en línia]. Estocolm, Suècia: 1968. Disponible a: <https://informs-sim.org/wsc68papers/1968_0054.pdf>. [Consulta: 15 juny 2020].

CALVO, Jorge. *¿Que es un Compilador en programación?* [en línia]. Madrid, España: 2018, 17 d'abril. Disponible a: <https://www.europeanvalley.es/noticias/que-es-un-compilador-en-programacion/>. [Consulta: 12 abril 2020].

CAR AND DRIVER, Redacción. *Simulación CFD y Túnel de Viento* [en línia]. Espanya: Hearst España S.L., 2007, 23 de juny. Disponible a: <https://www.caranddriver.com/es/formula-1/a15396/simulacion-cfd-y-tunel-de-viento/>. [Consulta: 19 febrer 2020].

CASSINI, Alejandro. *Modelos científicos* [en línia]. Buenos Aires, Argentina: editorial, 2015. Disponible a: <http://dia.austral.edu.ar/Modelos_cient%C3%ADficos>. [Consulta: 14 febrer 2020].

CASTRO, Iván; PÉREZ, Jesús Hernando. *LA GRAN REVOLUCIÓN ARITMÉTICA DE LA EDAD MEDIA Y EL SURGIMIENTO DEL ÁLGEBRA* [en línia]. Bogotà, Colòmbia: 2002. Disponible a: <https://www.redalyc.org/pdf/499/49925490001.pdf>. [Consulta: 19 maig 2020].

CUNA, Felipe. "*Pacific Blue", el ordenador más potente del mundo* [en línia]. 1998, 3 29 d'octubre. Disponible a: <https://www.elmundo.es/navegante/98/octubre/29/pacificblue.html>. [Consulta: 20 juny 2020].

DELOREAN. *Experimentos de Galileo* [en línia]. 2019, 09 de març. Disponible a: <https://es-la.facebook.com/828879200624515/posts/experimentos-de-galileocomo-segundo-tema-en-esta-serie-de-publicaciones-se-habla/1162506927261739/>. [Consulta: 16 juny 2020].

ET AL, Allen. *Chapter 5 What is discrete event simulation, and why use it?* [en línia]. Southampton, Anglaterra. Disponible a: <https://www.ncbi.nlm.nih.gov/books/NBK293948/>. [Consulta: 15 juny 2020].

FUNDACIÓN CEDE. *AMERICAN MANAGEMENT ASSOCIATION- AMA* [en línia]. Barcelona, Espanya: 2013. Disponible a: <http://www.directivoscede.com/es/convenio/american-management-association-ama>. [Consulta: 15 juny 2020].

GARCÍA, Bibiana; ARIAS, Daniel. *Al-Juarismi, puente matemático entre civilizaciones* [en línia]. Espanya: 2019, 04 de març. Disponible a: <https://www.bbvaopenmind.com/ciencia/matematicas/al-juarismi-puente-matematico-entre-civilizaciones/>. [Consulta: 17 maig 2020].

GARCIA, Ivan. *Primera generacion (1945-1956)* [en línia]. 2012, 27 de setembre. Disponible a: <https://sites.google.com/site/historiadelainformaticagarcia/primera-generacion>. [Consulta: 19 juny 2020].

GARCÍA, Julio. *Qué es Ruby y sus características* [en línia]. Espanya: OpenWebinars S.L., 2017, 20 d'octubre. Disponible a: <https://openwebinars.net/blog/que-es-ruby/>. [Consulta: 25 juliol 2020].

GARRIGA, Albert. *Método de Montecarlo en proyectos* [en línia]. Catalunya, Espanya: 2015, 02 de juliol. Disponible a: <https://www.recursosenprojectmanagement.com/metodo-de-montecarlo/>. [Consulta: 23 juny 2020].

GROUP, VirtualExpo. *Software para informes* [en línia]: *HyperWorks Virtual Wind Tunnel*. Independence. Disponible a: <https://www.directindustry.es/prod/altair/product-5874-1761027.html>. [Consulta: 19 febrer 2020].

HISTORIA Y VIDA. *¿Qué aportó a la ciencia Alan Turing?* [en línia]. Espanya: La Vanguardia, 2019, 12 de setembre. Disponible a: <https://www.lavanguardia.com/historiayvida/historia-contemporanea/20180611/47312986353/que-aporto-a-la-ciencia-alan-turing.html>. [Consulta: 25 febrer 2020].

HUMAN BRAIN PROJECT. *Short Overview of the Human Brain Project* [en línia]. 2020. Disponible a: <https://www.humanbrainproject.eu/en/about/overview/>. [Consulta: 15 juny 2020].

IBÁÑEZ, Juan José. *Concepto y Tipos de Modelos Científicos* [en línia]. Madrid: editorial, 2008, 10 de maig. Disponible a: <https://www.madrimasd.org/blogs/universo/2008/05/10/91441>. [Consulta: 14 febrer 2020].

IDICT, Carlos. *Fortran* [en línia]. Cuba: EcuRed, 2019, 19 d'agost. Disponible a: <https://www.ecured.cu/Fortran>. [Consulta: 15 juny 2020].

IDICT, Carlos. *CERN (Organización Europea para la Investigación Nuclear)* [en línia]. Cuba: EcuRed, 2019, 28 de juliol. Disponible a: <https://www.ecured.cu/CERN_(Organizaci%C3%B3n_Europea_para_la_Investigaci%C3%B3n_Nuclear)>. [Consulta: 25 juny 2020].

INIA. *Simulador de Cultivos* [en línia]. Montevideo, Uruguai: editorial, 2020, 24 de febrer. Disponible a: <http://www.inia.uy/gras/Alertas-y-herramientas/simulador-de-cultivos>. [Consulta: 25 febrer 2020].

JIMÉNEZ, Johel. *C#. Qué es y para qué se utiliza* [en línia]. Madrid, Espanya: 2018, 06 d'agost. Disponible a: <https://negociosyestrategia.com/blog/que-es-csharp/>. [Consulta: 25 juliol 2020].

KASHIWAMOTO, Eiso Jorge. *El concepto de CIBERNÉTICA en el Mundo Actual* [en línia]. Mèxic: 2017. Disponible a: <https://ingenieria.lasalle.mx/el-concepto-de-cibernetica-en-el-mundo-actual/>. [Consulta: 15 juny 2020].

LACASA, María Isabel. *Animacions i simulacions per a l'ensenyament de la Biologia a secundària* [en línia]. Sant Cugat del Vallès, Catalunya: 2007. Disponible a: <https://ddd.uab.cat/pub/ciencies/16996712n8/16996712n8p33.pdf>. [Consulta: 23 març 2020].

La construcción y uso de los modelos en las Ciencias Naturales y su Didáctica [en línia]. Colòmbia: Editorial Magisterio, 2016, 12 de juliol. Disponible a: <https://www.magisterio.com.co/articulo/la-construccion-y-uso-de-los-modelos-en-las-ciencias-naturales-y-su-didactica>. [Consulta: 14 febrer 2020].

LEGANEWS. *La UC3M crea un simulador informático que recrea la propagación del coronavirus* [en línia]. Leganés, Espanya: 2020, abril. Disponible a: <https://www.leganews.es/la-uc3m-crea-un-simulador-informatico-que-recrea-la-propagacion-del-coronavirus/>. [Consulta: 22 maig 2020].

LÓPEZ, Marisol. *Saps què és un algoritme?* [en línia]. Catalunya, Espanya: 2015, 04 de maig. Disponible a: <https://culturadigital.blog.gencat.cat/2015/05/04/saps-que-es-un-algoritme-2/>. [Consulta: 03 abril 2020].

LÓPEZ, Olivia. *Seis cosas que quizá no sabías de los algoritmos* [en línia]. Madrid: Ediciones EL PAÍS, s.l., 2019, 16 de febrer. Disponible a: <https://retina.elpais.com/retina/2019/02/06/album/1549450488_222524.amp.html>. [Consulta: 19 maig 2020].

MATEUS, Enrique. *El Cálculo Infinitesimal* [en línia]. Bogotà, Colòmbia: 2012. Disponible a: <https://edumatth.weebly.com/el-caacutelculo-infinitesimal.html>. [Consulta: 18 juny 2020].

MECAFENIX, Ingeniería. *¿Qué es un transistor y como funciona?* [en línia]. Mèxic: 2020, 03 de juny. Disponible a: <https://www.ingmecafenix.com/electronica/el-transistor/>. [Consulta: 20 juny 2020].

MECATRÓNICALATAM. *ÁLGEBRA BOOLEANA* [en línia]. Espanya: 2020. Disponible a: <https://www.mecatronicalatam.com/es/tutoriales/teoria/algebra-booleana/>. [Consulta: 19 juny 2020].

MELENDEZ, Rafael. *Lenguajes de Programación de Alto Nivel* [en línia]. Espanya: , 2019. Disponible a: <https://siaguanta.com/c-tecnologia/lenguajes-de-programacion-de-alto-nivel/>. [Consulta: 03 abril 2020].

MOROTE, José Luis. *Programas para la simulación energética de edificios* [en línia]. Espanya: 2016, 13 de gener. Disponible a: <https://ovacen.com/programas-para-la-simulacion-energetica-de-edificios/>. [Consulta: 25 maig 2020].

MUÑOZ, Ramón. *Informatica ambiente windows* [en línia]. Caracas, Veneçuela: 2016, 10 de juliol. Disponible a: <https://es.slideshare.net/RamonMuoz3/informatica-ambiente-windows>. [Consulta: 23 març 2020].

MUY INTERESANTE. *Una simulación muestra cómo se expande el coronavirus en un supermercado* [en línia]. Espanya: 2020, 14 d'abril. Disponible a: <https://www.muyinteresante.es/salud/articulo/una-simulacion-muestra-como-se-expande-el-coronavirus-en-un-supermercado-901586849383>. [Consulta: 22 maig 2020].

NATIONAL MUSEUM OF AMERICAN HISTORY. *IAS Computer* [en línia]. Washington, D.C. Disponible a: <https://americanhistory.si.edu/collections/search/object/nmah_334741>. [Consulta: 25 febrer 2020].

PASCUAL, Juan Antonio. *Los lenguajes de programación más populares del mundo* [en línia]. Espanya: Computer Hoy, 2020, 09 de maig. Disponible a: <https://computerhoy.com/listas/industria/lenguajes-programacion-mas-populares-633547>. [Consulta: 22 juliol 2020].

PASTOR. *La Aguja de Buffon* [en línia]. Navarra, Espanya: 2008. Disponible a: <.http://www.estadisticaparatodos.es/taller/buffon/buffon.html#:~:text=La%20Aguja%20de%20Buffon,lo%20que%20mide%20su%20di%C3%A1metro.&text=Una%20de%20ellas%20es%20el,Conde%20de%20Buffon%20%C2%BB%20en%201777.>. [Consulta: 15 juny 2020]

PE, Bruno. *¿Cómo crear un programa informático?* [en línia]. Perú: 2016, 03 de octubre. Disponible a: <https://bruno.pe/como-crear-un-programa-informatico/>. [Consulta: 12 abril 2020].

PEREZ, Jenny. *HISTORIA DE LA SIMULACIÓN* [en línia]. Londres, Anglaterra: 2020. Disponible a: <https://www.timetoast.com/timelines/historia-de-la-simulacion-42f0d38f-33d2-4980-991d-f2ca53ecf4fb>. [Consulta: 01 maig 2020].

PÉREZ, José Ángel. *Los modelos de la ciencia* [en línia]. Monterrey, Mèxic: editorial, 2016, 13 de gener. Disponible a: <https://monitor.iiiepe.edu.mx/notas/los-modelos-de-la-ciencia>. [Consulta: 14 febrer 2020].

PÉREZ, José Ángel. *Los modelos de la ciencia* [en línia]. Monterrey, Mèxic: editorial, 2016, 13 de gener. Disponible a: <http://www.3dcadportal.com/tunel-de-viento-virtual-de-hyperworks-de-altair-para-una-simulacion-aerodinamica-mas-rapida-y-exacta.html>. [Consulta: 14 febrer 2020].

RAFFINO, María Estela. *Concepto de TRANSISTOR* [en línia]. Argentina: 2020, 13 de juny. Disponible a: <https://concepto.de/transistor/>. [Consulta: 20 juny 2020].

RAMÍREZ, Iván. *Máquinas virtuales: qué son, cómo funcionan y cómo utilizarlas* [en línia]. Mèxic: 2020, 31 de gener. Disponible a: <*https://www.xataka.com/especiales/maquinas-virtuales-que-son-como-funcionan-y-como-utilizarlas*>. [Consulta: 17 agost 2020].

REDIGIT. *IRQ (INTERRUPCIONES DE HARDWARE): DEFINICIÓN Y SIGNIFICADO* [en línia]. Osca, Aragó: 2020. Disponible a: <https://blog.redigit.es/irq-interrupciones-de-hardware-definicion-y-significado/>. [Consulta: 16 juny 2020].

REPETTO, Juan Manuel. *Los simuladores (del agro)* [en línia]. Buenos Aires, Argentina: 2020, 25 de març. Disponible a: <http://sobrelatierra.agro.uba.ar/los-simuladores-del-agro/>. [Consulta: 25 febrer 2020].

ROBLEDANO, Ángel. *Qué es C++: Características y aplicaciones* [en línia]. Espanya: OpenWebinars S.L., 2019, 22 de juliol. Disponible a: <https://openwebinars.net/blog/que-es-cpp/>. [Consulta: 25 juliol 2020].

ROLDÁN, Ángel. *Historia de la Programación* [en línia]. Ciberaula: 2019. Disponible a: <https://www.ciberaula.com/cursos/java/historia_programacion.php>. [Consulta: 04 maig 2020].

ROUSE, Margaret. *MySQL* [en línia]. Nova York, E.E.U.U.: TechTarget S.A., 2015, gener. Disponible a: <https://searchdatacenter.techtarget.com/es/definicion/MySQL#:~:text=MySQL%20es%20un%20sistema%20de,incluyendo%20Linux%2C%20UNIX%20y%20Windows.>. [Consulta: 25 juliol 2020].

ROYANO, Ismael. *Cobol.¿Que es Cobol?* [en línia]. 2016, 11 de maig. Disponible a: <https://medium.com/enredando-con-programacion/cobol-que-es-cobol-3f86fa3a4394>. [Consulta: 25 juliol 2020].

RUIZA, M.; FERNÁNDEZ, T.; TAMARO, E.. *Blaise Pascal* [en línia]. Barcelona, Espanya: Biografías y vidas, 2004. Disponible a: <https://www.biografiasyvidas.com/biografia/p/pascal.htm>. [Consulta: 15 juny 2020].

RUIZA, M.; FERNÁNDEZ, T.; TAMARO, E.. *Gottfried Wilhelm Leibniz* [en línia]. Barcelona, Espanya: Biografías y vidas, 2004. Disponible a: <https://www.biografiasyvidas.com/biografia/l/leibniz.htm>. [Consulta: 15 juny 2020].

RUIZA, M.; FERNÁNDEZ, T.; TAMARO, E.. *Charles Babbage* [en línia]. Barcelona, Espanya: Biografías y vidas, 2004. Disponible a: <https://www.biografiasyvidas.com/biografia/b/babbage.htm>. [Consulta: 15 juny 2020].

RUIZA, M.; FERNÁNDEZ, T.; TAMARO, E.. *Norbert Wiener* [en línia]. Barcelona, Espanya: Biografías y vidas, 2004. Disponible a: <https://www.biografiasyvidas.com/biografia/w/wiener.htm>. [Consulta: 15 juny 2020].

RUIZA, M.; FERNÁNDEZ, T.; TAMARO, E.. *Konrad Zuse* [en línia]. Barcelona, Espanya: Biografías y vidas, 2004. Disponible a: <https://www.biografiasyvidas.com/biografia/z/zuse.htm>. [Consulta: 15 juny 2020].

SIGNIFICADOS. *Significado de Hardware* [en línia]. Espanya: 2019, 10 de juliol. Disponible a: <https://www.significados.com/hardware/>. [Consulta: 15 maig 2020].

SIGNIFICADOS. *Significado de Software* [en línia]. Espanya: 2019, 01 d'agost. Disponible a: <https://www.significados.com/software/>. [Consulta: 15 maig 2020].

SIMULART. *ProModel* [en línia]. Santiago, Chile: 2013. Disponible a: <http://www.simulart.cl/software-de-simulacion/software-promodel/>. [Consulta: 29 març 2020].

STEVENS, Harry. *Por qué brotes como el del coronavirus crecen exponencialmente y cómo "aplanar la curva"* [en línia]. Washington, D.C.: The Washington Post, 2020, 14 de març. Disponible a: <https://www.washingtonpost.com/graphics/2020/world/corona-simulator-spanish/>. [Consulta: 16 juny 2020].

TORRES, Karyna. *Cinta magnética* [en línia]. 2011, 27 de juliol. Disponible a: <https://sites.google.com/site/hardwareconceptosbasicos/Inicio/ci>. [Consulta: 19 juny 2020].

UNIVERSITARIS. *Un computador didáctico elemental (CODE-2)* [en línia]. Granada, Espanya: 2002. Disponible a: <http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2002/Cac141_148.pdf>. [Consulta: 10 juny 2020].

USEIT. *Llenguatges de programació, què són?* [en línia]. Lleida, Espanya: 2018, 21 de maig. Disponible a: <https://www.useit.es/ca/blog/llenguatges-de-programacio-que-son>. [Consulta: 20 juliol 2020].

WINTER SIMULATION CONFERENCE. *WSC Programs with Full Papers* [en línia]. Estats Units: 2019. Disponible a: <https://informs-sim.org/>. [Consulta: 15 juny 2020].

W3SCHOOLS. *HTML Element Reference* [en línia]. 2020. Disponible a: <https://www.w3schools.com/tags/default.asp>. [Consulta: 26 juliol 2020].

ANNEXOS

CODI FONT DEL MEU TREBALL

```
<head>

<title>Treball pràctic</title>

<style type="text/css">


.bg-image{
	background-image: url("img/fons.jpg");
	filter:blur(4px);
	-webkit-filter:blur(4px);
	height:100%;
	background-position:center;
	background-repeat:no-repeat;
	background-size:cover;
	}

#pista{
	position:absolute;
	background-color:darkgreen;
	border:60px solid grey;
	border-radius:100%;
	width:300px;
	height:300px;
	bottom:10px;
	margin-left:15vw;
	background:url(img/rotonda.png);
	background-repeat:no-repeat;
	background-size:cover;
	}

#coor{
	position:absolute;
	top:0px;
	left:0px;
	width:150px;
	height:150px;
	}

#centrar{
	position:absolute;
	top:50%;
	left:50%;
	width:600px;
	height:400px;
	margin-top:-200px;
	margin-left:-300px;
	}

#cotxe{
	position:absolute;
	width:60px;
	z-index:10000;
	bottom:0px;
	left:300px;
	}
```

```
h1{
	background-color:indianred;
	font-style:bold;
	border-style:double;
	border-width:5px;
	}

h2{
	position:absolute;
	font-family:verdana;
	top:0px;
	left:160px;
	font-size:42px;
	color:#f40404;
	font-weight:lighter;
	border-bottom-style:double;
	border-color:black;
	border-width:3px;
	}

#left{
	position:absolute;
	top:20%;
	left:5%;
	}

#boto{
	border:inset darkgray;
	border-width:7px;
	border-radius:5px;
	}

#buto{
	border:inset darkgray;
	border-width:7px;
	border-radius:5px;
	}

#velocitat{
	border-radius:10px;
	padding:15px;
	font-style:italic;
	font-weight:bold;
	font-family:verdana;
	}

#fregament{
	border-radius:10px;
	padding:15px;
	font-style:italic;
	font-weight:bold;
	font-family:verdana;
	}

input::placeholder{
	font-size:0.75em;
	}
```

```
#f1_logo{
	position:absolute;
	top:15px;
	left:15px;
	width:120px;
	}

#insti_logo{
	position:absolute;
	top:60px;
	right:60px;
	width:100px;
	border:4px solid black;
	}

</style>

<script type="text/javascript">
t=0;
R = 150;
function simular(){
T=t/10;

	v = document.getElementById("velocitat").value;
	f = document.getElementById("fregament").value;

//alert("Velocitat: "+velocitat);
//alert("Fregament: "+fregament);

w = v/R;
a = f*9.8-(v**2/R);
x = R+180*Math.cos(w*T);
y = 180*Math.sin(w*T);

if(a>0){
document.getElementById("cotxe").style.left = x+"px";
document.getElementById("cotxe").style.bottom = y+"px";
}else{
//clearInterval(grijo);
//alert("pum: "+a);
R = 150+0.5*a*T**2;
document.getElementById("cotxe").style.left = x+"px";
document.getElementById("cotxe").style.bottom = y+"px";
}

if(y>200 || x>350 || x<-150 || y<-300){
clearInterval(grijo);
//alert("todo no ok");
t=0;
document.getElementById("cotxe").src = "img/pum.gif";
}

t++;
if(t>5000){clearInterval(grijo);}
}
```

```
diode = 0;
function correr(){
if(diode == 1){
clearInterval(grijo);
diode = 0;
t=0;
}else{
grijo = setInterval("simular()",25);
diode = 1;
}
}

</script>

</head>


<body>

<div class="bg-image"></div>
<h2>Circuit de Vilamajor</h2>
<img id="f1_logo" src="img/f1_logo.png">
<img id="insti_logo" src="img/insti_logo.jpg">
<div id="left">
<h1>A màxima velocitat</h1>

<audio src="audio.mp3" preload="none" controls></audio>
<div>
      <hr>

      <input type="text" id="velocitat" placeholder="Introduir
velocitat..."></input><br>
      <br>
      <input type="text" id="fregament" placeholder="Introduir
coeficient de fregament..."></input><br>
      <br>
      <input type="button" id="boto" value="Click"
onclick="correr();"></input>
      <input type="button" id="buto" value="Reset"
onclick="window.location.reload(true)"></input>
</div>
</div>

<div id="centrar">
<div id="pista">
<div id="coor">
<img id="cotxe" src="img/cotxe.png">
</div>
</div>
</div>

</body>
```

RESULTAT VISUAL DE LA SIMULACIÓ

Enllaç: http://simulacio.rf.gd/?i=1