Simulasi Gerak Proyektil menggunakan GAViewer

Hafizh Afkar Makmur-13514062[1]
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13514062@std.stei.itb.ac.id

***Abstract*— Gerak Proyektil adalaha gerak sebuah objek menuju ke atas dengan sudut tertentu terhadap panah. Gerak ini bisa dianalisis dengan menggunakan vektor untuk mendeteksi setiap pergerakan dan setiap titik. Analisis ini bisa disimulasi menggunakan perangkat lunak GAViewer yang merupakan sebuah simulator 3D.**

***Keywords*— Animasi, GAViewer, Gerak Proyektil, Vektor**

I. PENDAHULUAN

Gerak proyektil merupakan salah satu persoalan paling mendasar dan menjadi salah satu topik yang pertama kali dibahas saat vektor diajarkan. Gerak ini dinamai dari gerak meriam yang biasanya dilambangkan dengan pergerakan sebuah objek (atau dalam kasus ini peluru) dengan kecepatan tertentu dari sebuah tempat di atas tanah menuju ke atas dengan membentuk sudut tertentu terhadap tanah. Beberapa persoalan yang selalu ditanyakan dalam gerak ini adalah mulai dari tinggi maksimum proyektil yang bisa dicapai, jarak maksimum yang bisa proyekril capai, atau sudut khusus sehingga sebuah titik bisa dicapai.

Salah satu cara paling mudah untuk memahami persoalan-persoalan itu adalah menggunakan analisis vektor. Gerak proyektil bisa dianalisis sebagai vektor untuk kemudian dilihat pengaruhnya terhadap gravitasi, gesekan udara, kecepatan angin, dan lain-lain. Jika gerak projektil sudah dianalisis seperti ini, posisi akhir maupun posisi di setiap waktu proyektil bisa diprediksi dengan tepat dan akurat menimbang bahwa semua masukan yang dibutuhkan tepat.

Gerak Proyektil sebagai salah satu model pertama siswa untuk mempelajari fisika membutuhkan model yang lebih mudah dipahami untuk mempermudah pemahaman menuju bidang-bidang studi fisika selanjutnya. Untuk itu, dibutuhkan sebuah simulasi bergerak yang bisa merepresentasikan gerak proyektil dengan baik dan menunjukkan gerakan setiap objek dengan lebih mudah.

GAViewer sebagai simulator 3D yang mendukung animasi mendukung pengembangan perangkat lunak untuk itu. Dengan pemrograman yang tepat, GAViewer dapat mensimulasi Gerak Proyektil dengan baik dan tepat. Makalah ini akan berfokus mengenai bagaimana menggunakan GAViewer untuk mensimulasikan gerak itu secara 2D dan 3D.

II. VEKTOR DALAM GERAK PROYEKTIL

A. Vektor dan Kecepatan

Vektor merupakan salah satu besaran geometri yang terdiri atas besar dan arah. Vektor banyak digunakan untuk memodelkan gerak benda-benda mulai dari benda yang berada di atas bumi sampai benda-benda langit seperti planet, bintang, dan asteroid. Vektor merupakan salah satu unsur paling penting dalam ilmu mekanika yang menganalisis bagaimana setiap benda bergerak dan bagaimana konsekuensinya dengan benda lain.

Pada mekanika, biasanya objek yang direpresentasikan dengan vektor adalah kecepatan. Besar kecepatan direpresentasikan dengan

$$v = \frac{s_t - s_0}{t}$$

dengan s_t adalah jarak objek di waktu tertentu dan s_0 adalah jarak awal. Rumus ini bisa dielaborasi lebih lanjut menjadi

$$v = \frac{s_t - s_{t-\Delta t}}{\Delta t}$$

untuk mendapatkan kecepatan sebuah objek di antara dua buah titik tertentu.

Untuk mendapatkan kecepatan sebuah objek tepat pada sebuah titik tertentu, berarti kita menginginkan untuk mendapatkan kecepatan dengan perbedaaan waktu 0. Pendekatan matematika biasa tidak mungkin digunakan untuk perhitungan itu karena akan melibatkan perhitungan 0/0 yang tidak terdefinisi. Untuk kasus itu, digunakan pendekatan kalkulus untuk menghasilkan rumus

$$\lim_{\Delta t \to 0} v = \frac{s_t - s_{t-\Delta t}}{\Delta t}$$

yang kemudian bisa dirubah menjadi

$$v = \frac{ds}{dt}$$

yang berarti kecepatan adalah turunan jarak terhadap waktu.

Dalam representasi vektor besar kecepatan menjadi besar vektor pada saat tertentu. Arah vektor kecepatan

ditentukan dari ke arah mana objek itu bergerak pada saat tertentu.

B.Akselerasi

Akselerasi dalam ilmu mekanika atau lebih tepatnya kinetika didefinisikan sebagai perubahan kecepatan sebuah objek dalam satuan waktu tertentu. Akselerasi secara matematis biasa dirumsukan sebagai

$$a = \frac{v_t - v_0}{t}$$

dengan v_t adalah kecepatan objek di waktu tertentu dan v_0 adalah kecepatan awal. Rumus ini bisa dielaborasi lebih lanjut menjadi

$$a = \frac{v_t - v_{t-\Delta t}}{\Delta t}$$

untuk mendapatkan akselerasi sebuah objek di antara dua buah titik tertentu.

Untuk mendapatkan akselerasi sebuah objek tepat pada sebuah titik tertentu, berarti kita menginginkan untuk mendapatkan akselerasi dengan perbedaaan waktu 0. Pendekatan matematika biasa tidak mungkin digunakan untuk perhitungan itu karena akan melibatkan perhitungan 0/0 yang tidak terdefinisi. Untuk kasus itu, digunakan pendekatan kalkulus untuk menghasilkan rumus

$$\lim_{\Delta t \to 0} a = \frac{v_t - v_{t-\Delta t}}{\Delta t}$$

yang kemudian bisa dirubah menjadi

$$a = \frac{dv}{dt}$$

yang berarti akselerasi adalah turunan kecepatan terhadap waktu.

Sebenarnya, akselerasi juga bisa direpresentasikan dengan vektor. Besar vektor akselerasi adalah besar akselerasi itu sendiri dengan arah vektor akselerasi bergantung pada arah vektor awal dan akhir dari akselerasi yang dihitung.

C.Gerak Lurus Berubah Beraturan

Gerak Lurus Berubah Beraturan terjadi pada sebuah objek yang bergerak lurus dengan kecepatan tertentu dan akselerasi konstan pada arah yang sama atau berkebalikan. Gerak ini merupakan pendahuluan dari studi gerak lain yang lebih kompleks.

Anggap ada sebuah objek dengan kecepatan awal atau v_0 sama dengan v. Pada waktu t, benda itu akan sudah menempuh jarak sebesar

$$s = v \, . \, t$$

Anggap jika benda sebelumnya memiliki akselerasi sebesar a dengan arah yang sama. Rumus jarak tidak akan menjadi sesederhana seperti yang sudah disebutkan di atas karena kecepatan tidak akan lagi menjadi konstan. Untuk kasus ini, kita bisa menggunakan anti-turunan menggunakan rumus yang ditulis di atas sehingga rumus kecepatan akhir adalah

$$v = v_0 + \int_0^t a \, dt$$

dan rumus kecepatan akhir menjadi

$$v = v_0 + at$$

Dengan demikian, rumus jarak total dapat dihitung sebagai

$$s = s_0 + \int_0^t v \, dt$$

dan rumus jarak total menjadi

$$s = s_0 + v_0 . t + \frac{1}{2} . a . t^2$$

D.Gerak Proyektil

Gerak Proyektil bisa dianalisis menjadi beberapa Gerak Lurus Berubah Beraturan yang terjadi secara bersamaan. Misalkan sebuah objek ditembakkan dengan kecepatan v dengan sudut sebesar θ terhadap tanah (sumbu x) dan sudut sebesar φ terhadap garis horizon (sumbu x). Tanpa percepatan gravitasi, kecepatan bisa dipecah menjadi tiga buah vektor v_x, v_y, dan v_z yang saling independen dengan besar masing-masing adalah

$$v_x = v \, . \cos(\theta) . \cos(\varphi)$$

$$v_y = v \, . \sin(\theta)$$

$$v_z = v . \cos(\theta) . \sin(\varphi)$$

Pada Gerak Proyektil sederhana, satu-satunya faktor eksteral yang mempengaruhi adalah faktor gravitasi yang merupakan vektor akselerasi yang memiliki besar sebesar g dan arah menuju sumbu y negatif. Berarti, kita mengetahui bahwa v_x dan v_z konstan dan hanya menyisakan v_y untuk diperhitungkan. Dengan mempertimbangkan faktor gravitasi, kecepatan objek di sumbu y di setiap waktu menjadi

$$v_y = v \, . \sin(\theta) - g . t$$

Pada rumus, terlihat bahwa jika v_y positif, lambat laun akselerasi gravitasi akan menurunkannya sedikit demi sedikit dan kemudian akan menjadikannya negatif. Perubahan ini terjadi secara linier. Namun, perubahan yang dilakukan gravitasi tidaklah linier. Jarak y atau s_y di waktu tertentu sebagaimana sudah dibahas pada bagian Gerak Lurus Berubah Beraturan dirumuskan sebagai berikut

$$s_y = s_{0y} + v.\sin(\theta).t - \frac{1}{2}.g.t^2$$

Berdasarkan rumus ini, lajur dari objek dalam sumbu y akan mengikuti persamaan kuadrat negatif. Itu berarti objek akan terlebih dahulu bergerak dengan cepat ke arah atas, kemudian mulai melambat sampai berhenti sesaat di puncak tertinggi, sampai akhirnya turun kembali dengan kecepatan yang semakin lama semakin cepat. Hal ini sesuai dengan perilaku objek yang dilempar ke atas di alam.

III. GAViewer dan Animasi

GAViewer merupakan software simulasi 3D yang bisa dikontrol baik dengan menggunakan *console* yang berada di software itu atau dengan menggunakan file berekstensi .g atau .geo. *Console* menyediakan sarana yang praktis untuk membuat berbagai objek secara cepat dan tepat namun tidak terlalu membuat nyaman untuk digunakan untuk membuat simulasi skala besar. File .g dan .geo biasanya digunakan untuk itu.

File .g memiliki struktur bahasa yang sangat mirip dengan bahasa C. Hal ini sangat memudahkan para pemrogram yang sudah terbiasa memrogram dengan bahasa ini. Adapun file .geo menggunakan strukturnya sendiri dengan format[1]

keyword arguments

dengan keyword berisi perintah apapun daengan argumen mereka masing-masing. File .geo tidak terlalu mirip dengan bahasa-bahasa lain namun memiliki cakupan yang lebih luas dari file .g maupun perintah konsol sehingga lebih baik untuk penggunaan GAViewer lebih lanjut.

Untuk membuat sebuah simulasi dinamis pada GAViewer, seseorang terlebih dahulu harus memahami bagaimana mendesain dan merias objek 3D statis pada GAViewer. Setelah itu, seseorang bisa menambah perintah untuk menggerakkan benda-benda itu sesuai dengan keinginan.

A.Paradigma Geometri GAViewer

GAViewer mampu menangani 3 jenis pendekatan perhitungan geometri yang ada. Ketiga model ini bisa diset di awal dengan menggunakan perintah default_model(<jenis model>) pada konsol. Ketiga model itu adalah[1] :

e3ga, ca3d : atau Euclidean Geometric Algebra. Model ini adalah model yang biasa digunakan dan diajarkan di pendidikan dasar sampai tinggi sebagai pengenalan. Model ini mudah dipahami dan mudah untuk direpresentasikan di dunia nyata namun belum bisa mencakup semua hal seperti bagaimana membentuk proyeksi vektor yang tidak berawal di titik 0,0, bagaimana memproyeksikan sebuah garis yang menghubungkan dua buah titik, dan lain-lain.

p3ga, ca4d : atau Projective Geometric Algebra, dan

c3ga, ca5d : atau Conformal Geometric Algebra. Model ini dianggap bisa mencakup semua simulasi geometri dan bisa menampung sampai 5 dimensi. Model ini biasanya baru diajarkan pada geometri tingkat lanjut dan pada [2] baru diberikan pada bab 11 dari 14. Semua **caNd** merupakan singkatan dari Clifford Algebra N Dimesional[1].

B. Vektor Basis

GAViewer adalah simulator 3 Dimensi sehingga untuk semua paradigma, vektor basis yang akan muncul secara nyata di layar hanya 3. Vektor basis ini dilambangkan dengan eN dengan N adalah dimensi ke-N. e1 melambangkan sumbu x. e2 melambangkan sumbu y, dan e3 melambangkan sumbu z. e4 dan e5 bisa digunakan namun tidak akan menampilkan apapun ke layar dan sepertinya digunakan untuk membantu perhitungan 5 Dimensi Conformal Geometry. no dan ni adalah vektor basis yang digunakan Conformal Geometry yang berarti Origin Vector dan Infinity Vector.

C. Simulasi 3D Statis

Beberapa komponen penting untuk membuat objek statis dalam GAViewer adalah bagaimana membuat titik, garis, vektor, dan objek. Hal ini bisa dilakukan baik pada konsol, file .g, ataupun file .geo. Kode yang harus dituliskan pada file .g dan konsol hampir mirip berbeda dengan kode pada file.geo.

Untuk membuat vektor, bidang, atau objek sederhana dan berpusat pada titik 0,0, kita bisa menggunakan paradigma Euclidean yang lebih mudah dipahami. Untuk membuat vektor, kita cukup meramu vektor basis menjadi sebuah persamaan vektor yang kita inginkan. Misalnya, untuk membentuk vektor sebesar 5.sqrt(3) dengan sudut θ sebesar 45^0 dan sudut φ sebesar 45^0, kita cukup menuliskan

```
v = 5*e1 + 5*e2 + 5*e3
```

Untuk membuat bidang, kita bisa membuat bivektor dengan format “parallelpiped without vector”. Bivektor dapat dibentuk menggunakan operator wedge (^) yang merupakan outer product dari dua buah vektor. Persamaan itu bisa berbentuk seperti ini

```
Alas = dm3(e1 ^ e2).
```

Persamaan ini akan membentuk sebuah bidang sebesar 1x1 yang berawal dari titik 0,0 menuju kuadran positif.

Untuk membentuk sebuah objek 3D, benda yang paling mudah dibuat adalah sphere atau bola. Benda ini bisa dibentuk dengan menggunakan penggunaan wedge seperti ini

```
Ball = e1 ^ e2 ^ e3
```

Untuk membentuk sebuah titik, vektor, bidang, ataupun

objek yang tidak berada di titik 0,0, harus digunakan pendekatan melalui c3ga atau Conformal Geometry. untuk membuat sebuah titik yang berada dikoordinat tertentu, persamaan yang harus dibentuk adalah

Point = c3ga_point(<vector>)

dengan titik itu akan seakan-akan berada di ujung vector yang berada di dalam c3ga_point.

Untuk membuat vektor, bidang, ataupun objek di luar titik 0,0, persamaan yang harus dibentuk menjadi

Object = vp(tv(tr + <vector>), no^(object equation))

dengan objek yang dimaksud akan menjadi seakan-akan berada di ujung vektor yang berada di dalam vp. Contoh dari penggunaan persamaan itu adalah

alas = vp(tv(tr -5*e1 -5*e3),no^(20*e1)^(10*e3))

Persamaan ini membentuk sebuah bidang sebesar 20x10 yang membentang pada sumbu xz dimulai dari koordinat (-5,0,-5).

Contoh lain yang berhubungan dengan vektor adalah

cvx = vp(tv(tr+(cx*e1)+(cy*e2)+(cz*e3)), no ^ (0.25*vx*e1))

Persamaan ini akan membentuk sebuah vektor sebesar vx pada sumbu x dengan titik awal di koordinat (cx,cy,cz).

Persamaan yang melibatkan objek bola 3D akan menjadi seperti ini

ball = vp(tv(tr+(cx*e1)+(cy*e2)+(cz*e3)), no ^ (0.1*e1) ^ (0.1*e2) ^ (0.1*e3)))

Persamaan ini akan membentuk sebuah bola sebesar 0.1*0.1*0.1 di koordinat (cx,cy,cz).

C. Simulasi 3D Dinamis

Ada dua komponen paling penting dalam pemrograman simulasi dinamis dalam GAViewer dan mereka adalah variabel atime dan dynamic statements. Variabel atime mencatat waktu yang sedang berjalan dan bisa memberi kita informasi yang baik untuk memanipulasi jalannya objek-objek di layar. Adapun dynamic statements menyatakan bahwa semua variabel yang disebut didalamnya dalah variabel dinamis yang saling bergantung satu sama lain. Hal ini berarti setiap ada nilai yang berubah dalam dynamic statements, semua nilai yang lain akan berubah mengikuti perubahan itu. Hal ini sangat berguna untuk membuat animasi karena hal ini menjamin setiap benda akan terus berubah seiring dengan waktu.

Ada dua perintah penting lain untuk menggerakan animasi dan mereka adalah start_animation() dan stop_animation. start_animation() mereset atime ke 0 dan kemudian menjalankannya. stop_animation() menghentikan pergerakan atime.

IV. SIMULASI DAN PERCOBAAN

A. Rincian Masalah

Pada simulasi ini dicoba untuk membuat sebuah objek (pada kasus ini sebuah bola) untuk bergerak sesuai dengan gerak proyektil yang sudah dibahas sebelumnya. Untuk kesempatan ini, satu-satunya faktor eksternal yang diperhatikan adalah gravitasi bumi yang saat ini ditetapakan g = 9.8 m//s^2. Itu berarti tidak ada gesekan udara ataupun angin yang akan mempengaruhi simulasi ini.

B. Desain Solusi

Untuk mensimulasikan gerak proyektil, dibuat sebuah obyek bernama “ball” yang berfungsi sebagai obyek yang akan bergerak proyektil. Masukan akan terdiri dari sudut awal pelemparan dan kecepatan awal. Kecepatan awal atau v nantinya akan dihitung proyeksinya terhadap sumbu x dan y dengan rumus yang sudah disebutkan di atas. Program akan mencatat waktu yang sedang berjalan dan kemudian me-*render* gambar awal. Saat gambar sudah di-*render*, program akan kembali mencatat waktu yang sedang berjalan dan membandingkannya dengan waktu sebelumnya. Setelah itu, program akan menggunakan posisi sebelumnya dan menghitung kecepatan di sumbu x dan y dan menghitung selisih waktu itu untuk mendapatkan posisi obyek saat ini. Setelah itu, program mencatat waktu yang sekarang dan kembali me-*render* gambar simulasi terbaru. Langkah ini akan terus diulang sampai program mendeteksi bahwa langkah berikutnya akan membuat objek akan menembus tanah (y < 0) dan kemudian animasi akan dihentikan.

C. Kode Sumber

Pada kesempatan ini, penulis menggunakan file .g untuk memprogram simulasi ini. Format .g digunakan karena kemiripannya dengan bahasa C sehingga lebih mudah dibaca dan dipahami dengan mudah oleh orang-orang dengan dasar Informatika.

Karena sulitnya memberi masukan pada konsol di GAViewer, diputuskan untuk memasukkan kecepatan awal, sudut awal, dan beberapa informasi lain langsung di dalam program.

```
1.  batch projectile() {
2.
3.      g = 9.81;                          // insert g
    ravity acceleration constant here
4.      v = 8;                             // insert v
    elocity here
5.      thetadeg = 45;                     // insert s
    tarting degree here (in degrees, between x and y)
6.      psideg = 30;                       // insert s
    tarting degree here (in degrees, between x and z)
7.      theta = thetadeg*pi/180;           // conversi
    on from degree to radian
8.      psi = psideg*pi/180;
9.      x = red(e1),
10.     y = green(e2),
11.     z = blue(e3),
12.     cx = 0;                            // inisiali
    asi cx, cy, dan cz
13.     cy = 0;
14.     cz = 0;
```

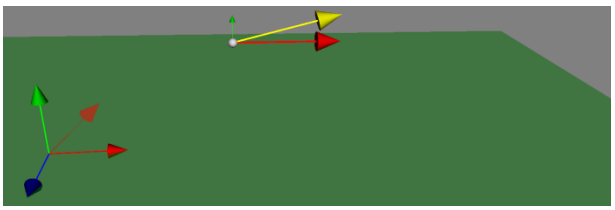
```
15.     ctime = atime;                          // inisiali
    asi ctime (current time)
16.     ctheta = theta;                         // inisiali
    sasi ctheta (current theta)
17.     alas = alpha(green(dm3(vp(tv(tr-5*e1-
    5*e3),no^(20*e1)^(10*e3)))),0.5),  // alas atau "ta
    nah"
18.     direction = red((v*cos(theta)*cos(psi)*e1 + v*s
    in(theta)*e2 + v*cos(theta)*sin(psi)*e3)*0.25),  //
     panah untuk menunjukkan arah awal
19.
20.     start_animation();      // mulai animasi
21.     dynamic { proj:
22.         ball = white(vp(tv(tr+(cx*e1)+(cy*e2)+(cz*e
    3)),no^(0.1*e1)^(0.1*e2)^(0.1*e3))), // posisi obje
    k
23.         cvx = red(vp(tv(tr+(cx*e1)+(cy*e2)+(cz*e3))
    ,no^(0.25*vx*e1))),                 // panah untuk
     vx pada objek
24.         cvy = green(vp(tv(tr+(cx*e1)+(cy*e2)+(cz*e3
    )),no^(0.25*vy*e2))),               // panah untuk
     vy pada objek
25.         cvz = blue(vp(tv(tr+(cx*e1)+(cy*e2)+(cz*e3)
    ),no^(0.25*vz*e3))),                // panah untuk
     vz pada objek
26.         cv = yellow(vp(tv(tr+(cx*e1)+(cy*e2)+(cz*e3
    )),no^(0.25*vx*e1+0.25*vy*e2+0.25*vz*e3))),      //
     panah untuk v pada objek
27.         vx = v*cos(theta)*cos(psi);
        // kecepatan pada sumbu x
28.         vy = v*sin(theta) -
     g*atime;    // kecepatan pada sumbu y (v*sin(theta
    ) - 1/2 *g*t^2)
29.         vz = v*cos(theta)*sin(psi);
        // kecepatan pada sumbu z
30.         ctheta = atan(vy/vx);                //
    theta sekarang
31.         cpsi = atan(vz/vx);                  //
    psi sekarang
32.         cx = cx + vx*(atime-
    ctime);             // posisi sekarang dalam sumbu
    x
33.         cy = cy + vy*(atime-
    ctime);             // posisi sekarang dalam sumbu
    y
34.         cz = cz + vz*(atime-
    ctime);             // posisi sekarang dalam sumbu
    z
35.         if (cy < 0) stop_animation();            //
     if the ball touch the ground, stop
36.         ctime = atime;                           //
    catat waktu sekarang
37.     }
38.
39. }
```

D. Beberapa Percobaan

Percobaan pertama dilakukan dengan masukan g = 9.81, v = 10, dan θ = 45 dan φ = 0. Percobaan ini dilakukan untuk mensimulasikan gerak proyektil 2D yang biasa dilakukan pada buku-buku fisika klasik.

Gambar 1.Percobaan 1 Awal

Bola berangkat dari titik 0,0 yang berada di sebelah kiri layar. Pada titik itu terlihat sumbu x, y, dan z yang berwarna masing-masing merah, hijau, dan biru. Panah merah putus-putus melambangkan arah bola akan bergerak sedangkan panah kuning menjadi bagian dari representasi v yang akan dijelaskan di bagian selanjutnya.



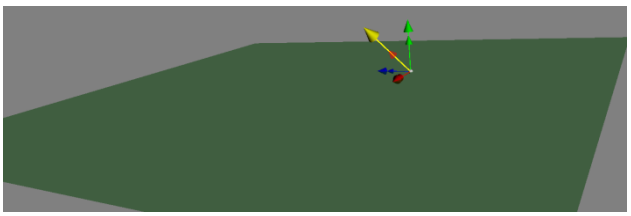
Gambar 2. Percobaan 1, Objek masih berada di udara

Program ini memberikan panah merah, kuning, dan hijau pada obyek untuk melambangkan v_x, v_y, dan v sekarang pada obyek. Seiring dengan bergeraknya obyek panah merah kuning dan hijau ini akan memanjang dan memendek sesuai dengan kecepatan bola itu di setiap titik.

Gambar 3. Percobaan 1, Obyek berhenti.

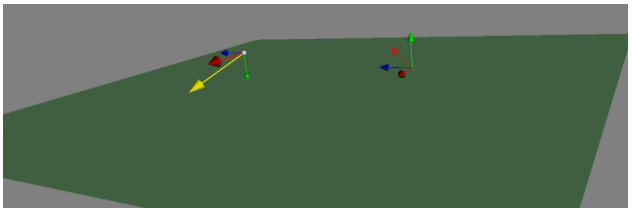
Seperti sudah disebutkan sebelumnya, objek akan berhenti tepat sebelum menyentuh tanah. Pada gambar di atas terlihat bahwa v_y, dan v mengarah ke bawah dengan besar yang kurang lebih sama besar dengan sebelumnya namun menunjuk ke arah bawah. Hal ini sesuai dengan hukum fisika yang sudah dijabarkan sebelumnya karena apa yang sudah diambil gravitasi pada waktu sebelumnya akan dikembalikan kembali dengan arah menuju ke bawah.

Percobaan selanjutnya adalah percobaan gerak proyektil 3D. Percobaan ini dilakukan menggunakan masukan g = 9.81, v = 10, dan θ = 45 dan φ = 30



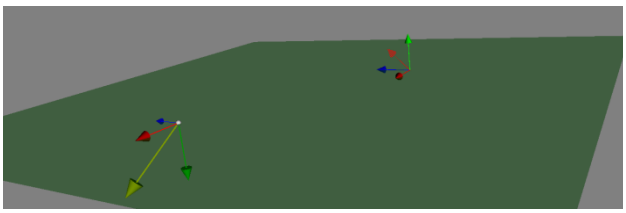
Gambar 4. Percobaan 2 Awal

.Pada percobaan ini, terlihat vektor baru berwarna biru yang berada di sampaing bola. Vektor ini melambangkan v_z yang tidak muncul pada percobaan sebelumnya karena saat itu proyeksi v terhadap z adalah 0.



Gambar 5. Percobaan 2, Objek berada di udara

Sebagaimana v_x, v_z konstan terhadap waktu, menyisakan v_y untuk berdansa dengan waktu.



Gambar 6. Percobaan 2 bola mencapai tanah

Seperti pada percobaan sebelumnya, kedudukan akhir vektor pada bola hampir sama persis dengan kedudukannya di awal, hanya membedakan arah v_y yang sekarang menunjuk ke arah bawah atau negatif dan arah v yang mengikuti gabungan v_x, v_y, dan v_z yang sudah berubah.

E. Kelebihan dan Kekurangan Program

Program ini sudah bisa mensimulasi gerak proyektil lengkap dengan informasi kecepatan pada sumbu x dan kecepatan pada sumbu y di setiap waktu. Program ini juga sudah bisa memberikan jarak maksimum obyek yang dilempar dengan melihat posisi bola saat menyentuh tanah. Terlebih lagi, program ini bisa mensimulasikan gerak proyektil 3D dengan memasukkan theta dan psi dengan tepat.

Program ini masih memiliki banyak kekurangan dalam memperjelas beberapa bagian seperti skala, informasi angka secara numerik dan *real-time*, serta penggambaran sudut seperti pada buku fisika standar berupa sebuah busur di antara sudut secara informasi besar sudur secara numerik dan *real-time*. Program ini juga belum menampilkan berbagai faktor lain seperti gesekan udara dan kecepatan angin yang tentu saja memengaruhi gerak objek di dunia nyata.

Pengembangan lebih lanjut masing mungkin untuk mengembangkan program ini lebih lanjut dengan menyelesaikan masalah-masalah di atas. Bisa direkomendasikan untuk menambahkan kasus-kasus lain seperti saat sasaran berada di posisi yang lebih tinggi atau lebih rendah dari posisi penembak.

V. KESIMPULAN

Dengan menggunakan sumber yang ada, gerak proyektil bisa disimulasikan dengan GAViewer. Untuk pembuatan simulasi yang lebih nyata dan akurat, diperlukan pengembangan lebih banyak lagi dan masukan yang lebih akurat lagi untuk membuat simulasi ini lebih akurat dan lebih mudah dipahami dan dipelajari.

VI. PERSEMBAHAN

Karya ini pertama kali dipersembahkan kepada Allah *Subhanahu wa Ta'ala* yang atas limpahan rahmat dan nikmatnya penulis mampu menyelesaikan makalah ini dengan baik.

Karya ini juga dipersembahkan pada dosen-dosen Aljabar Geometri yang sudah memperkenalkan penulis pada dunia yang baru dan menyenangkan ini. Semoga mereka terus diberi kesehatan dan kelancaran untuk membagi ilmu yang bermanfaat.

DAFTAR PUSTAKA

Fontijne, Daniel, "GAViewer Documentation Version 0.85,," : University of Amsterdam, 2010

Vince, John, "Geometric Algebra for Computer Sciences" : Springer-Verlag London Limited 2008

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 16 Desember 2015

Hafizh Afkar Makmur - 13514062