BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Implementasi merupakan tahap penciptaan perangkat lunak dan juga tahap kelanjutan dari perancangan aplikasi. Selanjutnya dilakukan uji kelayakan terhadap aplikasi yang telah dibangun sehingga dapat memberikan masukan demi berkembangnya aplikasi yang telah dibangun. Tahap ini merupakan tahap dimana aplikasi siap untuk dioperasikan, tahap ini terdiri dari penjelasan mengenai lingkungan implementasi, baik itu lingkungan perangkat keras, lingkungan perangkat lunak, serta implementasi aplikasi.

4.1.1 Implementasi Perangkat Keras

Perangkat keras yang digunakan untuk menjalankan simulasi pembelajaran fisika momentum dan impuls dapat dilihat pada tabel 4.1.

Tabel 4.1 Tabel Spefisikasi Perangkat Keras

No	Perangkat Keras	Keterangan
1	*Processor*	Kecepatan @ 1.83GHz (2 CPUs)
2	*Memory* (RAM)	1536MB
3	*Display Memory*	256 MB
4	*Display Mode*	1280 x 800 (32 bit) (60Hz)
5	Keyboard	Standard
6	Mouse	Standard

4.1.2 Implementasi Perangkat Lunak

Adapun spesifikasi perangkat lunak yang digunakan untuk menjalankan simulasi pembelajaran fisika momentum dan impuls dapat dilihat pada tabel 4.2.

Tabel 4.2 Tabel Spefisikasi Perangkat Lunak

No	Perangkat Lunak	Spesifikasi
1	Sistem Operasi	Windows 7
2	*Tools Compiler*	Adobe Flash CS4

4.1.3 Implementasi Aplikasi

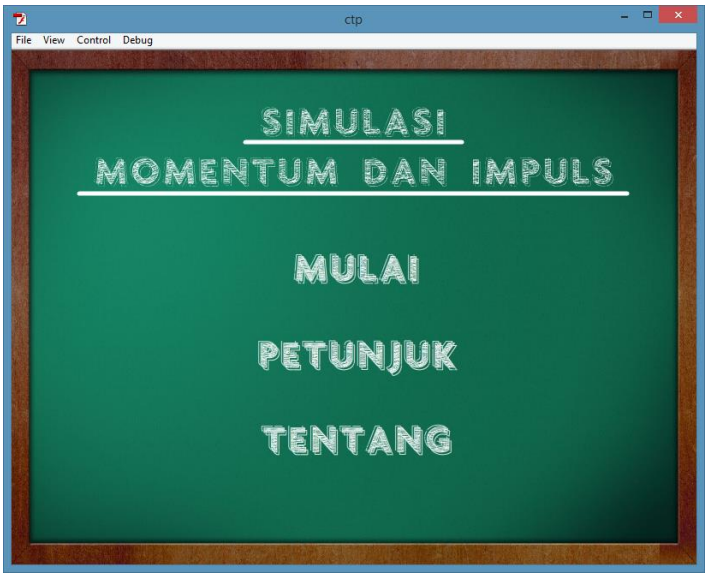
Aplikasi simulasi pembelajaran fisika momentum dan impuls merupakan sebuah simulasi yang bertjuan menggambarkan kondisi yang terjadi diantara dua buah benda saat terjadi momentum dan impuls sebagai sarana pembelajaran bagi siswa yang merupakan implementasi dari algoritma *gilbert jhonson keerthi*. simulasi pembelajaran fisika momentum dan impuls ini dapat dijalankan pada sistem operasi windows.

4.1.4 Implementasi Antarmuka

Antarmuka merupakan sarana interakasi antara pengguna dengan sistem operasi. Berikut ini merupakan antarmuka yang ada pada simulasi pembelajaran fisika momentum dan impuls :

1. Antarmuka Menu Utama

Antarmuka menu utama merupakan tampilan awal ketika aplikasi dijalankan. Tampilan antarmuka menu utama dapat dilihat pada gambar 4.1.



Gambar 4.1 Antarmuka Menu Utama

2. Antarmuka Mulai Simulasi

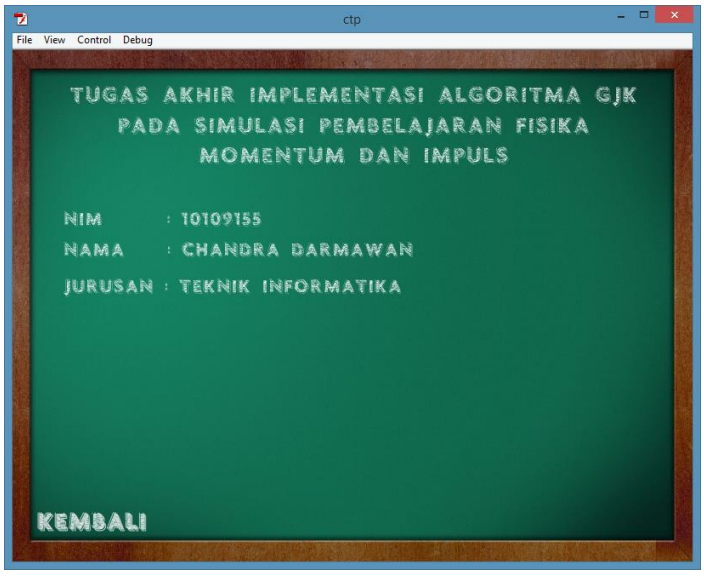
Antarmuka mulai simulasi merupakan antarmuka untuk menampilkan simulasi dari algoritma *gilbert jhonson keerthi*. Tampilan antarmuka mulai permainan dapat dilihat pada gambar 4.2.



Gambar 4.2 Antarmuka Mulai Simulasi

3. Antarmuka Tentang

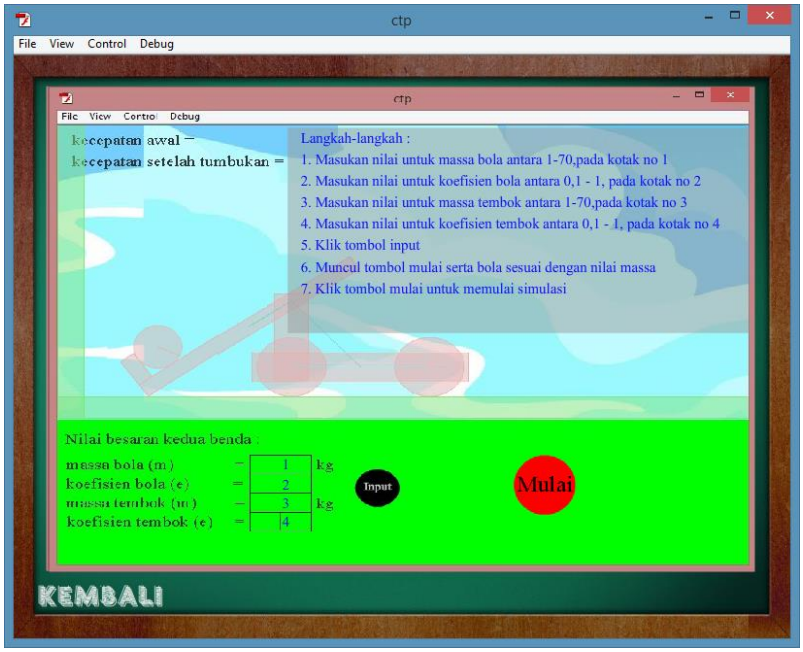
Antarmuka Tentang merupakan tampilan tentang pembuat program. Tampilan antarmuka tentang dapat dilihat pada gambar 4.3.



Gambar 4.3 Antarmuka Tentang

4. Antarmuka Petunjuk

Antarmuka Petunjuk merupakan tampilan yang bertujuan memberikan penjelasan dasar menggunakan simulasi Tampilan antarmuka petunjuk dapat dilihat pada gambar 4.4.



Gambar 4.4 Antarmuka Petunjuk

4.2 Pengujian

Pengujian bertujuan untuk menemukan kekurangan atau kesalahan pada perangkat lunak, serta untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan perancangan perangkat lunak tersebut. Adapun pengujian yang akan dilakukan adalah pengujian alpha.

4.2.1 Pengujian Alpha

Pengujian alpha merupakan pengujian fungsional yang digunakan untuk menguji sistem. pengujian alpha hanya untuk sirkulasi internal dan masalah *(error)* atau ketidak lengkapan yang terdapat dalam aplikasi. Proses pengujian terhadap aplikasi akan menggunakan pengujian *black box* dan pengujian *white box*.

4.2.1.1 Skenario Pengujian

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada simulasi pembelajaran fisika momentum dan impuls. Adapun skenario pengujian yang ada pada simulasi pembelajaran fisika momentum dan impuls dapat dilihat pada tabel 4.3.

Tabel 4.3 Tabel Skenario Pengujian

No	Kelas Uji	Butir Uji	Jenis Pengujian
1	Algoritma Gilbert Jhonson Keerthi	Uji algoritma tumbukan dua *convex shape*	*White box*
2	Menu Utama	Memilih Mulai Simulasi	*Black box*
		Memilih Petunjuk	*Black box*
		Memilih Tentang	*Black box*
3	Mulai Simulasi	Ketika mimilih tombol mulai	*Black box*
		Tombol input variable nilai benda	*Black box*
		Tombol mulai simulasi	*Black box*
		Tombol ulang	*Black box*
		Tombol kembali	*Black box*
4	Petunjuk	Ketika memilih tombol petunjuk	*Black box*
		Tombol kembali	*Black box*
5	Tentang	Ketika memilih tombol tentang	*Black box*
		Tombol kembali	*Black box*

4.2.1.2 Pengujian *White Box*

Pengujian *white box* merupakan metode perancangan *test case* yang menggunakan struktur kontrol dari perancangan metode dalam mendapatkan *test case*. Adapun metode yang digunakan dalam pengujian *white box* ini adalah metode *basis path*. Metode *basis path* mengijinkan pendesain kasus uji untuk mendapatkan perkiraan lojik yang kompleks dari desain metode dan menggunakan perkiraan ini untuk mendefinisikan aliran eksekusi.

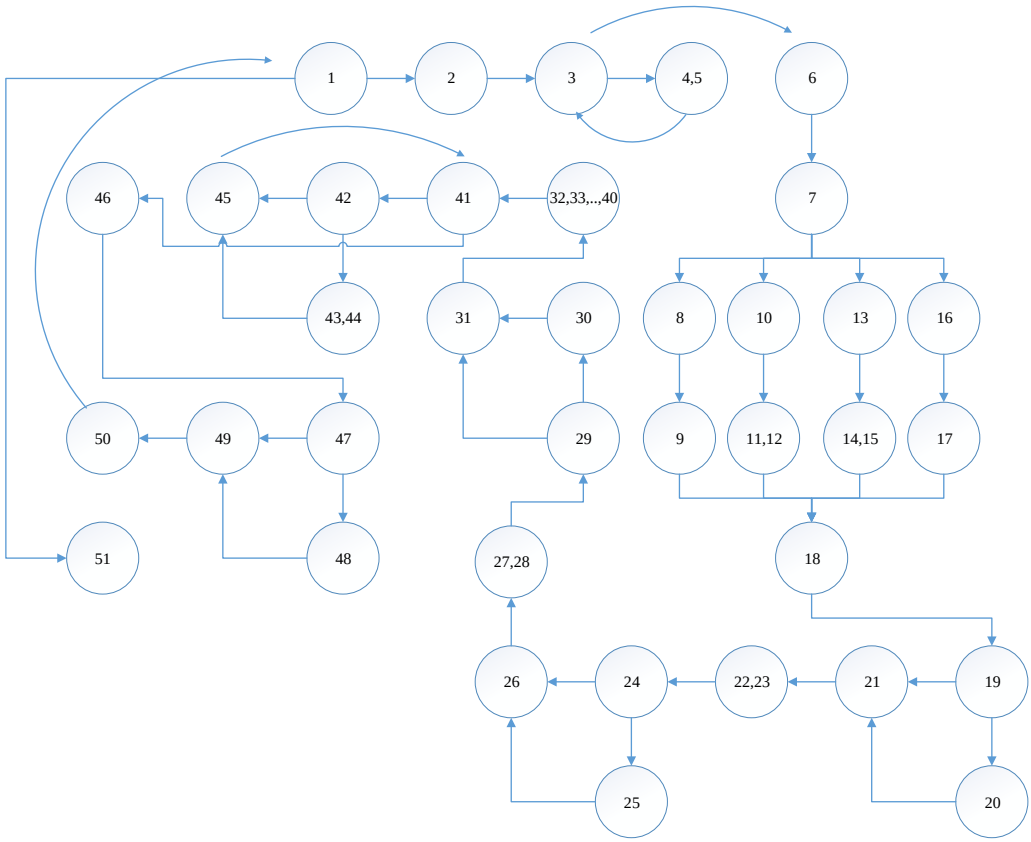
Tabel 4.4 Pseudocode Algoritma Gilbert Jhonson Keerthi

No	**Pseudocode**
1	while (iter < k_maxIters) {
2	saveCount = simplex.m_count;
3	for (i = 0; i < saveCount; i++){
4	saveA[i] = vertices[i].indexA;
5	saveB[i] = vertices[i].indexB;
6	}
7	switch(simplex.m_count){

```
8   case 1:
9   break;
10  case 2:
11  simplex.Solve2();
12  break;
13  case 3:
14  simplex.Solve3();
15  break;
16  default:
17  b2Settings.b2Assert(false);
18  }
19  if (simplex.m_count == 3){
20  break;
21  }
22  p = simplex.GetClosestPoint();
23  distanceSqr2 = p.LengthSquared();
24  if (distanceSqr2 > distanceSqr1){
25  break;
26  }
27  distanceSqr1 = distanceSqr2;
28  var d:b2Vec2 = simplex.GetSearchDirection();
29  if (d.LengthSquared() < Number.MIN_VALUE * Number.MIN_VALUE){
30  break;
31  }
32  var vertex:b2SimplexVertex = vertices[simplex.m_count];
33  vertex.indexA = proxyA.GetSupport(b2Math.MulTMV(transformA.R,
    d.GetNegative()));
34  vertex.wA = b2Math.MulX(transformA, proxyA.GetVertex(vertex.indexA));
35  vertex.indexB = proxyB.GetSupport(b2Math.MulTMV(transformB.R, d));
36  vertex.wB = b2Math.MulX(transformB, proxyB.GetVertex(vertex.indexB));
37  vertex.w = b2Math.SubtractVV(vertex.wB, vertex.wA);
38  ++iter;
39  ++b2_gjkIters;
40  var duplicate:Boolean = false;
41  for (i = 0; i < saveCount; i++){
42  if (vertex.indexA == saveA[i] && vertex.indexB == saveB[i]){
43  duplicate = true;
44  break;
45  }
46  }
47  if (duplicate){
48  break;
49  }
50  ++simplex.m_count;
51 }
```

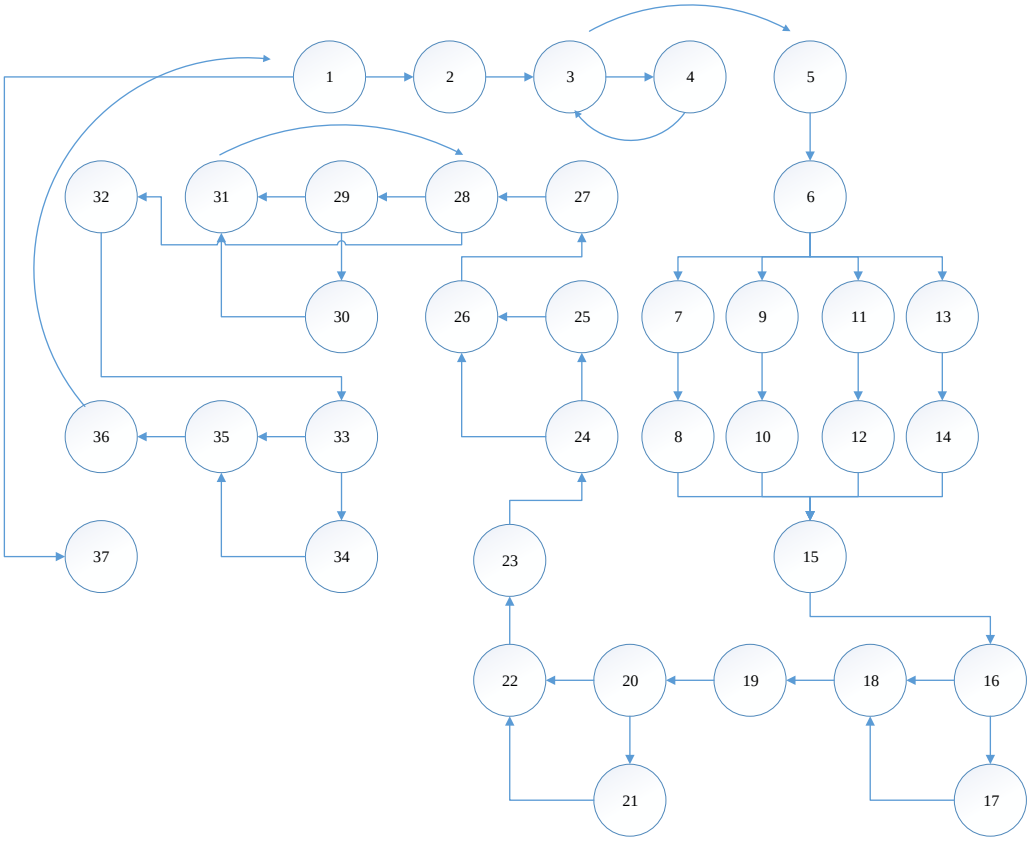
a. *Flow Graph*

Berdasarkan *pseudocode* diatas maka *flow graph* dari algoritma gilbert jhonson keethi dapat dilihat pada gambar 4.5.



Gambar 4.5 *Flow Graph* algoritma GJK

atau disederhanakan berdasarkan simpul dapat dilihat pada gambar 4.6.



Gambar 4.6 Penyederhanaan *Flow Graph* algoritma GJK

Keterangan :

= *Nodes* (N)

= *Edges* (E)

Berdasarkan gambar 4.6 maka dapat dihitung *cyclomatic complexity* sebagai berikut :

a. *Cylomatic Complexity* V(G)

V(G) = E – N + 2

V(G) = 47 – 37 + 2

V(G) = 12

Keterangan :

E = Jumlah edge pada grafik alir

N = Jumlah node pada grafik alir

b. *Independent Path*

Path 1 = 1-2-3-4-3-5-6-7-8-15-16-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

Path 2 = 1-2-3-4-3-5-6-9-10-15-16-17-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

Path 3 = 1-2-3-4-3-5-6-11-12-15-16-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

Path 4 = 1-2-3-4-3-5-6-13-14-15-16-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

Path 5 = 1-2-3-4-3-5-6-7-8-15-16-17-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

Path 6 = 1-2-3-4-3-5-6-7-8-15-16-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

Path 7 = 1-2-3-4-3-5-6-7-8-15-16-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

Path 8 = 1-2-3-4-3-5-6-7-8-15-16-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

Path 9 = 1-2-3-4-3-5-6-7-8-15-16-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

Path 10 = 1-2-3-4-3-5-6-7-8-15-16-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

Path 11 = 1-2-3-4-3-5-6-7-8-15-16-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

Path 12 = 1-2-3-4-3-5-6-7-8-15-16-18-19-20-22-23-24-26-27-28-29-30-31-28-32-33-34-35-36-1-37

c. *Graph Matrix*

Berikut adalah *graph matrix* yang menyatakan hubungan antara *nodes* dan *edges* pada gambar 4.6. Kolom *Sum*-1 yang merupakan nilai dihitung berdasarkan jumlah node yang ada dikurang 1. Jika node dari busur selanjutnya tidak ada, maka dinyatakan sebagai elemen kosong dan bernilai nol.

Tabel 4.5 *Graph Matrix*

NO	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	Sum-1
1		1																																			1	1
2			1																																			
3				1	1																																	1
4				1																																		
5						1																																
6							1		1		1		1																										3
7								1																														
8															1																								
9										1																												
10															1																								
11												1																										
12															1																								
13														1																								
14															1																								
15																1																							
16																	1	1																					1
17																		1																					
18																			1																				
19																				1																			
20																					1	1																	1
21																						1																	
22																							1																
23																								1															
24																									1	1													1
25																										1													
26																											1												
27																												1											
28																													1			1							1
29																														1	1								1
30																															1								
31																												1											
32																																1							
33																																		1	1				1
34																																			1				
35																																				1			
36	1																																						
37																																							
Jumlah (Sum-1) + 1																																							12

Keterangan:

1. Baris dan kolom merepresentasikan simpul (*node*)
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul.

Kesimpulan :

Dari hasil pengujian *white box* dapat dilihat bahwa jumlah *edges* (E) pada *graph matrix* sama dengan jumlah *cyclomatic complexity* yaitu 12, maka dapat disimpulkan bahwa proses aturan *alignment* berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

d. Pengujian Algoritma Gilbert Jhonson Keerthi

Tabel 4.6 Pengujian Algoritma Gilbert Jhonson Keerthi

No	Kasus Uji	Hasil Yang Diharapkan	Hasil Uji	Keterangan
1	Memberikan nilai k_maxIters >= 20 while(iter<k_maxIters){ …}	Perulangan tidak dilakukan (Berhenti)	k_maxIters >= 20 Perulangan tidak dilakukan (Berhenti)	[√] Alur Terlewati [ ] Alur Tidak Terlewati

2	for (i = 0; i < saveCount; i++){
	
	saveA[i]= vertices[i].indexA;
	
	saveB[i]= vertices[i].indexB;}
	
	switch(simplex.m_count){
	
	case 1:
	
	break;
	
	case 2:
	
	simplex.Solve2();
	
	break;
	
	case 3:
	
	simplex.Solve3();
	
	break;
	
	default:
	
	b2Settings.b2Assert(false);}
		Mengkopi *simplex* sehingga dapat mengindifikasi duplikasi	Simplex tersimpan	[√] Alur Terlewati
				
				[ ] Alur Tidak Terlewati

3	Memberikan nilai simplex poin = 3
	
	if (simplex.m_count == 3){
	
	break;
	
	}
		Jika simplex bernilai 3 poin maka *origin* adalah di segitiga	Simplex berada pada *origin*	[√] Alur Terlewati
				
				[ ] Alur Tidak Terlewati

4	p= simplex.GetClosestPoint() ;
	
	distanceSqr2= p.LengthSquared();
	
	if(distanceSqr2 > distanceSqr1){
	
	//break;}
	
	distanceSqr1= distanceSqr2;
		Menghitung jarak poin terdekat	Simplex berada pada point mendekati *origin*	[√] Alur Terlewati
				
				[ ] Alur Tidak Terlewati

5	if (d.LengthSquared() < Number.MIN_VALUE * Number.MIN_VALUE){ break;}	Mencari arah pencarian	Arah pencarian baru ditemukan	[√] Alur Terlewati [ ] Alur Tidak Terlewati

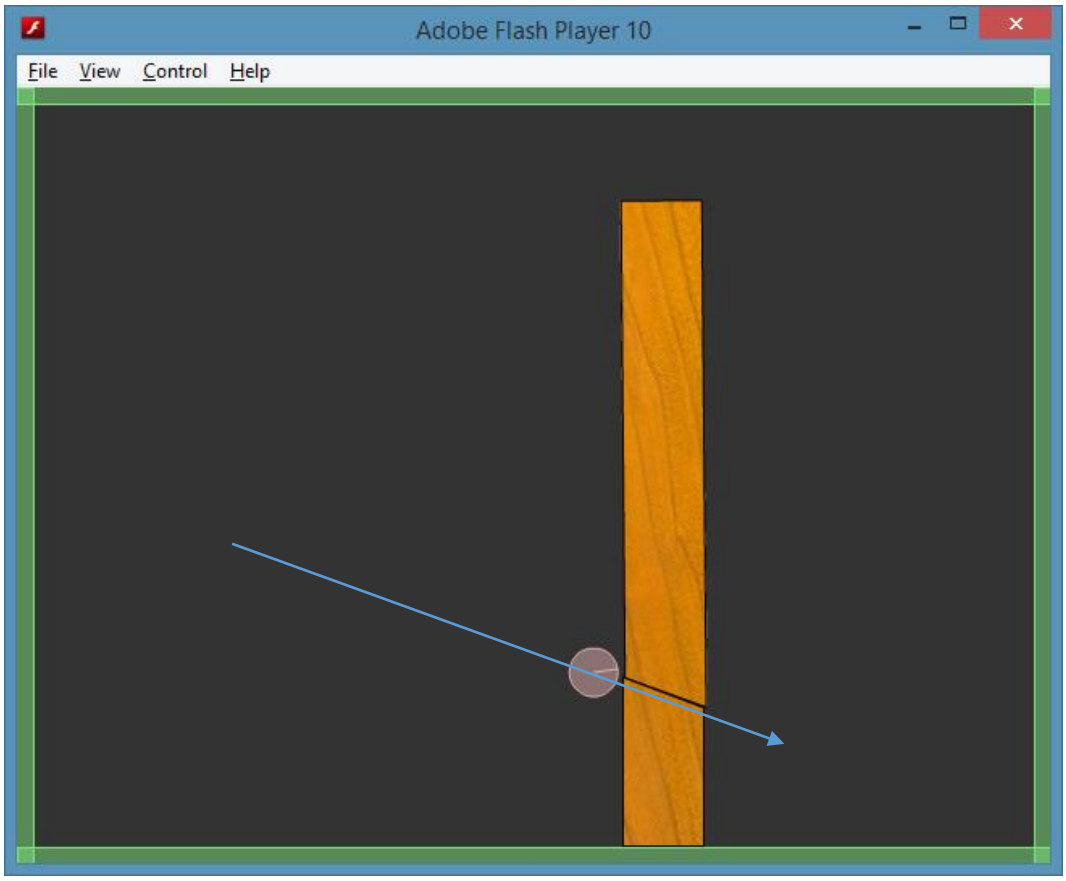
6	for (i = 0; i < saveCount; i++){
	
	if (vertex.indexA == saveA[i] && vertex.indexB == saveB[i]){
	
	duplicate = true;
	
	break;}}
		Mengecek *support point* jia terjadi duplikasi	*Support point* mengindetifikasi duplikasi *vertex.*	[√] Alur Terlewati
				
				[ ] Alur Tidak Terlewati

7	if (duplicate){ break;}	Jika menemukan duplikasi, support point menghindari *cycling*		[√] Alur Terlewati [ ] Alur Tidak Terlewati

8	++simplex.m_count;	Menciptakan simplex baru.	Simplex baru tercipta sesuai arah pencarian	[√] Alur Terlewati [ ] Alur Tidak Terlewati

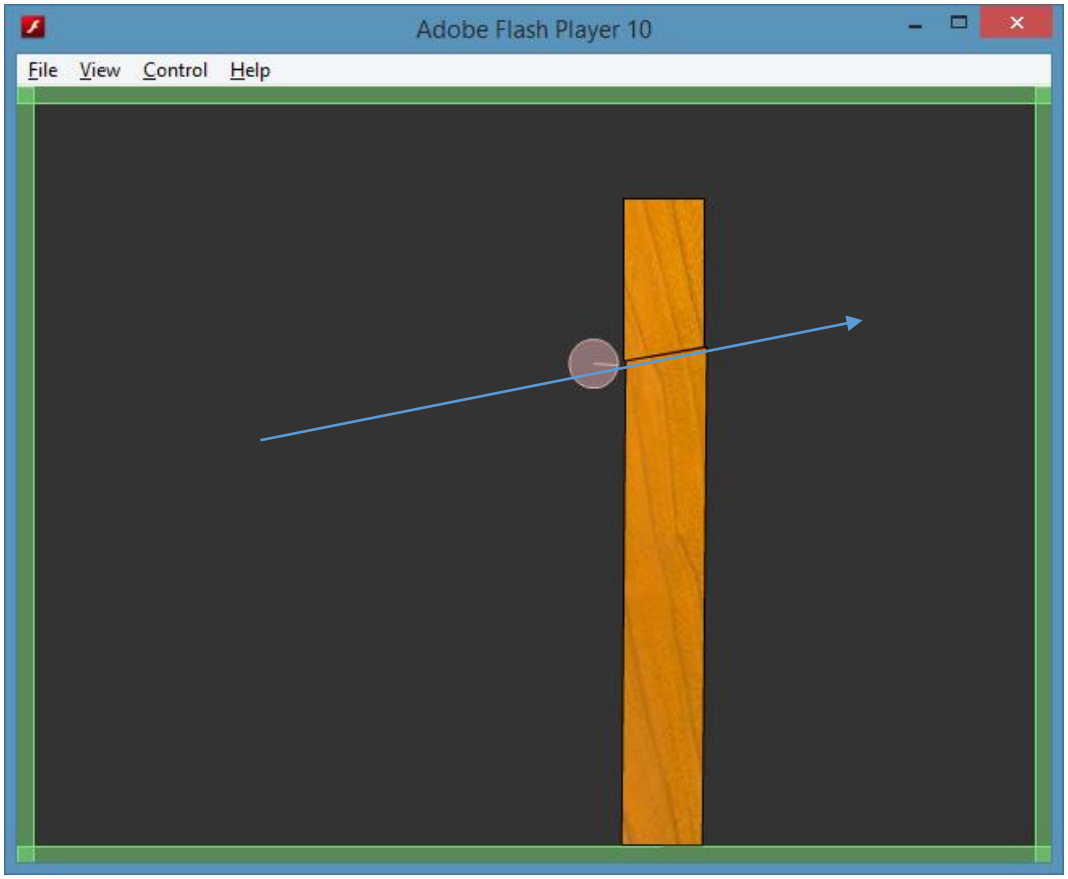
4.2.2 Pengujian Algoritma *Gilbert Jhonson Keerthi* Pada Simulasi Fisika Momentum dan Impuls

Pengujian dilakukan untuk membuktikan bahwa algoritma *GJK* telah bekerja sesuai rancangan. Pengujian ini dilakukan dengan menguji titik tumbukan dan hasil keakuratan arah *impact* yang terjadi setelah terjadi impuls.



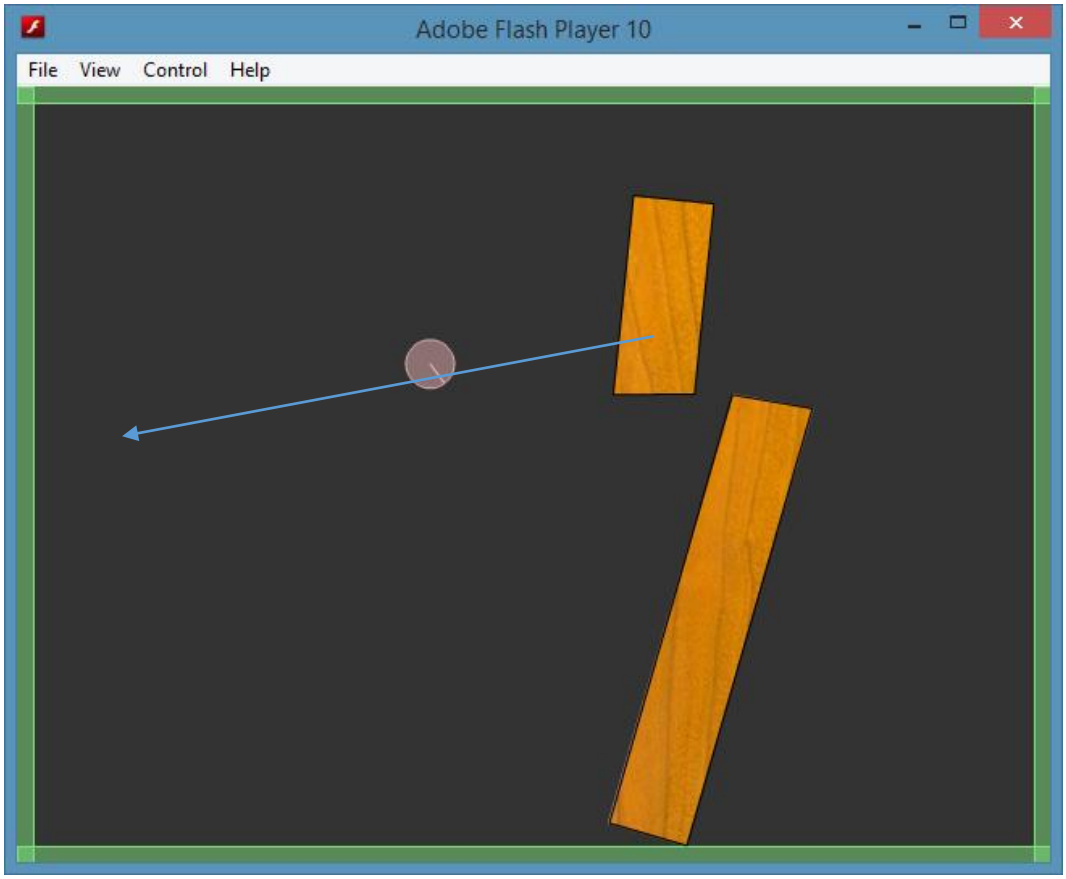
Gambar 4.7a Pengujian titik temu convex hull bola terhadap papan

Pengujian ini akan membuktikan bahwa algoritma GJK mendeteksi titik temu *convex hull* pada saat tumbukan sesuai rancangan.



Gambar 4.7b Pengujian titik temu convex hull bola terhadap papan

Pengujian ini akan membuktikan bahwa algoritma GJK mendeteksi titik temu *convex hull* pada saat tumbukan sesuai rancangan.



Gambar 4.8 Pengujian *impact* bola terhadap papan setelah terjadi impuls

Pengujian ini akan membuktikan bahwa algoritma GJK memberikan dampak terhadap arah yang ditimbulkan setelah terjadi impuls sesuai dengan titik temu *convex hull.*

4.2.1.1 Pengujian *Black Box*

Pengujian *fungsional* yang digunakan untuk menguji sistem yang baru adalah metode pengujian alpha. Metode yang digunakan dalam pengujian ini adalah pengujian *black box* yang berfokus pada persyaratan fungsional dari sistem yang dibangun.

Kasus dan Hasil Pengujian

1. Pengujian Menu Utama

Tabel 4. 6 Pengujian Menu Utama

Kelas Uji	Skenario Uji	Hasil Yang Diharapkan	Kesimpulan
Menu Utama	Memilih Tombol Mulai Simulasi	Menampilkan Area Simulasi	[√] Berhasil [ ] Tidak Berhasil
	Memilih Tombol Petunjuk	Menampilkan Petunjuk	
	Memilih Tombol Tentang	Menampilkan Tentang	

2. Pengujian Mulai Simulasi

Tabel 4. 7 Pengujian Mulai Simulasi

Kelas Uji	Skenario Uji	Hasil Yang Diharapkan	Kesimpulan
Mulai Simulasi	Ketika Memilih tombol mulai	Menampilkan Area Simulasi	[√] Berhasil [ ] Tidak Berhasil
	Tombol input variable nilai benda	Menampilkan bola dengan variabel yang telah ditentukan	
	Memilih Tombol Mulai Simulasi	Mengeksekusi bola dengan catapult	
	Memilih Tombol Ulang	Mengembalikan kondisi simulasi seperti semula	
	Memilih Tombol Kembali	Kembali ke menu utama	

3. Pengujian Menu Petunjuk

Tabel 4. 8 Pengujian Menu Petunjuk

Kelas Uji	Skenario Uji	Hasil Yang Diharapkan	Kesimpulan
Menu Petunjuk	Ketika Memilih Menu Petunjuk	Menampilkan Petunjuk	[√] Berhasil [ ] Tidak Berhasil
	Memilih Tombol Kembali	Kembali ke menu utama	

4. Pengujian Menu Tentang

Tabel 4. 9 Pengujian Menu Tentang

Kelas Uji	Skenario Uji	Hasil Yang Diharapkan	Kesimpulan
Menu Tentang	Ketika Memilih Menu Tentang	Menampilkan Tentang	[√] Berhasil [ ] Tidak Berhasil
	Memilih Tombol Kembali	Kembali ke menu utama	

4.2.2 Pengujian Beta

Pengujian beta dilakukan dengan tujuan untuk mengetahui sejauh mana kualitas dari perangkat lunak yang dibangun, apakah sudah sesuai dengan harapan atau belum. Untuk itu dalam pengujian beta dilakukan penelitian dengan cara melakukan wawancara langsung dengan Drs.Sukandar selaku guru fisika di SMA Pasundan 3 Cimahi sebagai pengguna. Dari hasil wawancara tersebut dapat diketahui penilaian calon pengguna terhadap perangkat lunak yang telah dibangun. Daftar pertanyaan dalam wawancara yang diberikan adalah sebagai berikut :

1. Dengan dibangunnya simulasi pembelajaran fisika momentum dan impuls di SMA Pasundan 3 Cimahi, apakah menambah fasilitas pendukung pembelajaran mata pelajaran fisika?

   Jawab: Setuju, karena dengan dibuatnya simulasi ini dapat mendukung siswa dalam mencoba mempraktekan kondisi momentum benda yang bergerak dengan perbedaan nilai besaran dalam kondisi parabola dan mampu mensimulasikan perubahan nilai besaran ketika terjadinya impuls ketika terjadi tumbukan terhadap rintangan yang muncul.

2. Apakah hasil impact yang disimulasikan aplikasi ini telah sesuai dengan parameter masukan bola?

   Jawab: Setuju, karena terlihat dari arah impact yang ditimbulkan setelah terjadi tumbukan sesuai dengan perbedaan massa benda serta koefisien restitusi bola tersebut.

3. Apakah titik temu convex hull antara dua buah *convex shapes* pada saat terjadi impuls, telah tersimulasikan secara tepat?

   Jawab : Setuju, karena arah benda saat titik temu convex hull antara dua buah convex shapes tersebut bertumbukan sesuai dengan titik berat benda sehingga simulasi lebih terlihat menggambarkan kondisi sebenarnya.

4.2.2.1 Kesimpulan Pengujian beta

Berdasarkan hasil pengujian beta, dapat disimpulkan bahwa :

1. Aplikasi simulasi pembelajaran fisika momentum dan impuls yang dibangun dapat menambah fasilitas pendukung mata pelajaran fisika di SMA Pasundan 3 Cimahi
2. Hasil simulasi sesuai dengan parameter masukan
3. Titik temu convex hull antara dua buah *convex shapes* pada saat terjadi impuls, telah terdeteksi dengan tepat.