# RANCANG BANGUN GAME EDUKASI MENGENAL BANGUN DATAR UNTUK ANAK PRA SEKOLAH

**SKRIPSI**

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Grandis Mahendra Wahyu Widodo
NIM: 105060800111071


INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2015

# PENGESAHAN

RANCANG BANGUN GAME EDUKASI MENGENAL BANGUN DATAR
UNTUK ANAK PRA SEKOLAH

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Grandis Mahendra Wahyu Widodo
NIM: 105060800111071

Skripsi ini telah diuji dan dinyatakan lulus pada
29 Oktober 2015
Telah diperiksa dan disetujui oleh:

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Eriq Muh. Adams J., S.T, M.Kom | Wibisono Sukmo Wardhono, S.T, M.T |
| NIP: 19850410 201212 1 001 | NIK: 201008 820404 1 001 |

Mengetahui
Ketua Program Studi Informatika/Ilmu Komputer

Marji, Drs., M.T
NIP: 19670801 199203 1 001

# PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 17 November 2015

Grandis Mahendra W W
NIM: 105060800111071

# KATA PENGANTAR

Puji syukur saya ucapkan kepada Allah SWT yang senantiasa melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Proposal Skripsi yang berjudul, "Rancang Bangun Game Edukasi Mengenal Bangun Datar Untuk Anak Pra Sekolah". Skripsi ini diajukan sebagai salah satu syarat untuk mendapatkan gelar sarjana S-1 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.

Melalui kesempatan ini, penulis menyampaikan rasa hormat dan terima kasih penulis yang sebesar-besarnya kepada pihak yang telah memberikan bantuan lahir maupun batin selama penulisan skripsi ini. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Bapak Eriq M. Adams J. S.T., M.Kom. dan Bapak Wibisono Sukmo Wardhono, S.T., M.T. selaku dosen pembimbing penulis. Terima kasih atas semua bimbingan dan dorongan semangatnya.
2. Bapak Ir. Sutrisno, M.T., Bapak Ir. Heru Nurwasito, M.Kom., Bapak Himawat Aditya, S.T., M.Sc. dan Bapak Edy Santoso, S.Si., M.Kom. selaku Ketua, Wakil Ketua 1, Wakil Ketua 2, Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Drs. Marji, M.Si. dan Issa Arwani, S.T., M.T. selaku Ketua dan Sekretaris Program Studi Informatika serta segenap Bapak/Ibu Dosen, Staff Administrasi dan Perpustakaan Program Studi Informatika Universitas Brawijaya.
4. Seluruh dosen Program Studi Informatika atas kesediaan membagi ilmunya kepada penulis.
5. Semua Asisten Ka. Lab serta Laboran dari Laboratorium Program Studi Informatika yang telah memberikan banyak bantuan dan dukungan dalam menyelesaikan skripsi ini.
6. Ayah Jonni Heri Widodo, S.Pd., Ibu Sri Wahyuni, S.Pd., kakak dan adik penulis serta seluruh keluarga yang senantiasa tiada henti hentinya memberikan do'a demi terselesainya skripsi ini.
7. Teman – teman Pandawa Lima (Firsta Bagus Sugiharto, S.Pd., M.Pd., Prayoga Purnama Putra, S.Pd., Bagus Purnomo Aji, S.Pd. dan Arik Achmad Efendy, S.Kom.) yang telah memberikan semangat untuk segera menyelesaikan skripsi ini.
8. Teman-teman *MIRACLE* (Afi Muftihul S., S.Kom., Albilaga Linggra P., S.Kom., Alvin Hermawan, S.Kom., Anas Rachmadi P., S.Kom., Andriyanto, S.Kom., Claudio Fresta S., S.Kom., Dwi Hardyanto, S.Kom., Dwi Vendy P., S.Kom., Fendy Gusta P., S.Kom., I Putu Yoga P., S.Kom., Nadia Previani R., S.Kom., Rosikhan Maulana Y., S.Kom., Sheila Zivana L., S.Kom., dan Weni Prameswari, S.Kom.,) yang telah memberikan dorongan dan bantuan baik secara moril maupun materiil demi lancarnya penyusunan tugas akhir ini dan semoga yang belum menyusul segera cepat menyusul.

9. Ibu Ifa selaku kepala sekolah "TK RA Muslimat NU 16 Malang" tempat penulis melakukan penelitian.
10. Teman-teman siswa "TK RA Muslimat NU 16 Malang" yang telah membantu penulis dalam penelitian dengan memainkan *game* edukasi yang telah dibuat.
11. Teman-teman Raion Studio yang telah memberikan inspirasi dalam pembuatan ide skripsi ini.
12. Teman – teman Raion Community yang telah berbagi ilmu dan pengalaman dengan penulis.
13. Serta semua pihak yang namanya tidak bisa penulis sebutkan satu persatu. Terima kasih atas do'a dan dukungannya.

Penyusun sadar bahwa masih banyak kesalahan dan kekurangan dalam penyusunan Skripsi ini, untuk itu penyusun mohon maaf dan mengharapkan kritik dan saran guna penyempurnaan selanjutnya.

Malang, 17 November 2015

Penulis

grandisgt@gmail.com

# ABSTRAK

Mengenal bentuk bangun datar merupakan salah satu mata pelajaran yang terdapat pada kurikulum pendidikan taman kanak-kanan (TK). Mata pelajaran ini diberikan untuk dasar kepada anak-anak sebelum memasuki usia sekolah. Di TK RA Muslimat NU 16 Malang materi mengenal bangun datar dijelaskan oleh guru dengan menggunakan papan yang berbentuk bangun datar seperti persegi, lingkaran, segitiga, dan jajar genjang. Kurangnya permainan dalam memberikan materi tersebut membuat anak tidak fokus dalam belajar, sehingga di akhir pembelajaran banyak anak yang masih bingung dalam menyebutkan bentuk-bentuk segi empat, segitiga, lingkaran dan jajar genjang saat mereka mengamati bentuk roda, bentuk rumah, bentuk layang-layang, dan bentuk yang ada di sekitar. Pembuatan media pembelajaran yang menarik dan interaktif diharapkan dapat meningkatkan kemampuan anak-anak untuk memahami berbagai bentuk bangun bangun datar dan dapat membentuk suatu bentuk baru dari kumpulan bentuk bangun datar dengan membuat sebuah aplikasi game edukasi.

*Game* dibangun dengan melalui tiga tahapan yaitu perancangan *game* dengan metode MDA *framework*, *prototype* dan implementasi. Implementasi *game* ini menggunakan bahasa pemrograman C# dan menggunakan *game engine* Unity 3D. Pengujian aplikasi ini menggunakan metode white-box testing dengan strategi pengujian *basis path*, pengujian *test flow diagram* menggunakan teknik jalur *minimum path generation* dan *focus testing.* Berdasarkan hasil pengujian terhadap pengguna (*Focus Testing*), beberapa pertanyaan yang diajukan mendapat tanggapan yang positif dari anak-anak TK yang memainkannya. Dari hasil pengujian didapatkan kesimpulan bahwa permainan ini telah memenuhi kebutuhan fungsional dan non fungsional.

Kata kunci: *game* edukasi, bangun datar, Unity 3D, *prototype,* MDA *framework,* anak pra sekolah

# ABSTRACT

*Knowing about two-dimentional figure is one of the subjects included in the educational curriculum for kindergarten. This subject are given for children basic knowledge before entering elementary school. TK RA Muslimat NU 16 Malang teachers introduce the 2D figure to children using board shaped like a square, circle, triangle, and parallelogram. Lack of game during teaching process make children do not focus in learning, so they are still confused to identified shape of rectangle, triangle, circle and parallelogram from real objects like a wheel, home, kite, etc. An educational game application is needed for making an interesting and interactive learning media that can improve children's ability to understand the 2d figure and make a new shape from collecting 2D figures.*

*This constructed in three stages, first game is designed with MDA framework, prototype and implementation. Implementation of this game is using C# as programming language and Unity 3D as game engine. This application testing is using unit testing with white-box testing method, test flow diagrams with minimum track path generation techniques and focus testing. Based on test ing results to the user (Focus Testing), some questions received a positive response from kindergarten children who play. From the test results it was concluded that this game has completed the functional and non-functional requirements.*

*Keywords: Education game, 2D figure, Unity 3D, prototype, MDA framework, kindergarten*

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB 1 PENDAHULUAN

## 1.1 Latar Belakang

Anak merupakan generasi muda yang menjadi investasi dan harapan masa depan bangsa dan bermain adalah sesuatu yang hampir tidak terlepaskan dari anak-anak. Sebagian besar waktu mereka adalah untuk bermain. Dalam kegiatan inilah sejatinya mereka banyak belajar. Mereka selalu ingin tahu dan mencoba. Bermain merupakan aktivitas yang dilakukan tanpa adanya tujuan yang serius. Satu-satunya tujuan adalah perasaan senang pada saat melakukannya. Bermain tidak sekedar mengisi waktu, tetapi merupakan kebutuhan anak seperti halnya makanan, perawatan dan cinta kasih. Selain itu bermain juga merupakan unsur yang penting untuk perkembangan fisik, emosi, mental, intelektual, kreativitas dan sosial (Soetjiningsih, 1995). Karenanya, bermain sangat dekat dengan kehidupan anak-anak.

Masa anak-anak dimulai setelah melewati masa bayi yang penuh ketergantungan. Masa anak-anak awal berlangsung dari usia 2 tahun sampai 6 tahun, oleh para pendidik dinamakan sebagai usia pra-sekolah. Pada usia prasekolah anak-anak akan mengalami perkembangan yang sangat cepat dari segi fisik, kognitif, emosi maupun sosial. Di masa inilah anak-anak perlu mendapatkan pendidikan yang disebut dengan pendidikan anak pra sekolah. Pendidikan anak pra sekolah adalah suatu upaya pembinaan yang ditujukan bagi anak dari usia 2 tahun sampai dengan usia 6 tahun yang dilakukan melalui pemberian rangsangan pendidikan untuk membantu pertumbuhan dan perkembangan jasmani dan rohani agar anak memiliki kesiapan dalam memasuki pendidikan lebih lanjut. Taman kanak-kanak (TK) sebagai lembaga pendidikan formal pertama merupakan salah satu sarana untuk membantu memberi rangsangan dan dukungan dalam masa pertumbuhan dan perkembangan anak (Taurena D, 2008).

Pada kurikulum pendidikan taman kanak-kanak (TK) terdapat mata pelajaran mengenai pengenalan bentuk geometri atau bangun datar. Mata pelajaran ini diberikan untuk dasar kepada anak-anak sebelum memasuki usia sekolah. Mengenal bentuk adalah langkah awal bagi anak-anak untuk mengamati, mempelajari, serta mengelompokkan apa yang mereka lihat. Karakter atau ciri yang mudah dikenal ini merangsang anak-anak untuk menjelaskan dan mengatur dunia di sekeliling mereka yang penuh dengan keragaman.

Di TK RA Muslimat NU 16 Malang materi mengenal bangun datar dijelaskan oleh guru dengan menggunakan papan yang berbentuk bangun datar seperti persegi, lingkaran, segitiga, dan jajar genjang. Berdasarkan bentuk papan tersebut, guru akan memberikan instruksi kepada anak-anak untuk mengelompokkan benda disekitar yang bentuknya sama dengan papan tersebut. Kurangnya permainan dalam mengenalkan bangun datar membuat anak tidak fokus dalam belajar, banyak anak yang mengobrol dengan temannya atau

bermain sendiri dan tidak memperhatikan guru. Sehingga di akhir pembelajaran banyak anak yang masih bingung dalam menyebutkan bentuk-bentuk segi empat, segitiga, lingkaran dan jajar genjang saat mereka mengamati bentuk roda, bentuk rumah, bentuk layang-layang, dan bentuk yang ada di sekitar. Begitu juga ketika diberi tugas dalam membentuk bentuk baru dari kumpulan bentuk bangun datar banyak anak yang masih merasa kesulitan.

Perkembangan teknologi yang sangat pesat dewasa ini memberikan efek yang dapat dirasakan secara langsung maupun tidak langsung. Salah satunya yang dapat dirasakan langsung adalah jarangnya permainan tradisional yang kini beralih menjadi permainan modern seperti video *game*. Saat ini video *game* menjadi salah satu bagian dalam kehidupan masyarakat modern dari anak kecil sampai dengan orang dewasa. Video *game* sendiri saat ini lebih banyak memberikan efek buruk pada anak-anak walaupun video *game* menjadi cabang olah raga elektronik atau disebut *eSport* dan di Indonesia baru diresmikan tgl 24 Juni 2014. Hal ini dikarenakan kurangnya *game* yang bertemakan edukasi. Berdasarkan kajian di atas maka dengan banyaknya orang yang memainkan *game* maka pengembangan *game* (*game development*) mulai mengarah ke dalam industri *game* edukasi. Hal ini dimaksudkan selain mendapatkan hiburan dalam bermain *game*, pemain juga mendapatkan nilai tambah yaitu pengetahuan (Martono, 2011).

Dalam Game Developers Converence di San Jose dengan judul acara Game Design and Tuning Workshop telah dikembangkan dan diajarkan MDA *framework*. MDA adalah pendekatan formal untuk memahami *game* dan salah satu upaya yang menjembatani kesenjangan antara *game design* dan *development, game critiscm* dan *technical game research.* Metodologi ini dipercayai dapat memperjelas dan memperkuat proses *iterative* dari para *developer,* pelajar dan peneliti yang membuat *game* sehingga memudahkan semua pihak untuk terurai dalam mempelajari dan merancang kelas yang luas dari *game design* dan *game artifact* (Hunicke, et al., 2004).

Dengan ini maka dibuatlah *game* edukasi mengenai pengenalan bangun datar untuk anak pra sekolah sebagai sarana pembelajaran yang menarik dan interaktif yang dapat meningkatkan kemampuan anak-anak untuk memahami berbagai bentuk bangun bangun datar dan dapat membentuk suatu bentuk baru dari kumpulan bentuk bangun datar dengan judul “Shape Adventure” dengan menggunakan MDA *framework*.

## 1.2 Rumusan Masalah

Berdasarkan paparan latar belakang tersebut, maka rumusan masalah adalah sebagai berikut:

1. Bagaimana merancang *game* “Shape Adventure”, sebagai sarana pembelajaran untuk anak pra sekolah dengan menggunakan MDA *framework*?

2. Bagaimana mengimplementasikan rancangan *game* edukasi "Shape Adventure" menjadi sebuah aplikasi *game* agar bisa diterima menjadi media pembelajaran?
3. Bagaimana mengetahui apakah *game* edukasi "Shape Adventure" berjalan sesuai dengan kebutuhan dan dapat diterima menjadi sarana pembelajaran yang menarik?

## 1.3 Batasan Masalah

Untuk menghindari adanya kemungkinan semakin berkembangnya masalah, maka penulisan laporan ini hanya menitikberatkan permasalahan pada beberapa hal, diantaranya adalah :

1. Penelitian akan dilakukan di TK RA Muslimat NU 16 Malang untuk anak kelas TK B, yaitu anak yang berusia 5 - 6 tahun.
2. Jenis-jenis bangun datar yang akan digunakan yaitu lingkaran, persegi, persegi panjang, segitiga, layang-layang, jajar genjang, dan trapesium.
3. Pembuatan *game* edukasi menggunakan *game engine* Unity 3D versi 4.6 dengan bahasa pemrograman C#.
4. *Game* yang akan dibangun menggunakan model dan animasi 2D.

## 1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk merancang *game* mengenal bangun datar untuk anak pra sekolah dengan menggunakan MDA *framework*, mengimplementasikannya menjadi sebuah aplikasi *digital,* dan mengetahui apakah *game* "Shape Adventure" berjalan sesuai dengan kebutuhan dan dapat diterima menjadi sarana pembelajaran yang menarik mengenal berbagai macam bentuk bangun datar.

## 1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Bagi Penulis
   - Dapat lebih memahami tentang pembuatan *game* edukasi untuk anak-anak khususnya anak di usia prasekolah.
2. Bagi Anak - anak
   - Dapat memberikan pembelajaran dengan permainan yang menarik.
   - Dapat memberikan kemudahan dalam belajar bangun datar kepada anak-anak yang belajar di bangku pra sekolah.
3. Bagi Pembimbing
   - Dapat menjadikan sebagai media pembelajaran.
   - Dapat mempermudah dalam pengajaran mengenal bangun datar.

## 1.6 Sistematika Penulisan

Untuk mencapai tujuan yang diharapkan, maka sistematika penulisan yang disusun dalam tugas akhir ini adalah sebagai berikut:

**BAB 1 Pendahuluan**

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

**BAB 2 Landasan Kepustakaan**

Membahas penelitian terkait dan teori-teori yang mendukung dalam pembuatan *game* edukasi mengenai bangun datar untuk anak pra sekolah.

**BAB 3 Metodologi Penelitian**

Membahas tentang metode yang digunakan dalam penulisan yang berupa diagram alir yang terdiri dari studi literatur, perancangan *game*, *paper prototype*, implementasi *game* serta pengujian *game*.

**BAB 4 Hasil**

Membahas tentang pengujian *game* yang dilakukan beserta metode yang digunakan dalam melakukan pengujian.

**BAB 5 Pembahasan**

Membahas tentang hasil dari pengujian *game* yang telah dilakukan.

**BAB 6 Penutup**

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian *game* edukasi yang dikembangkan dalam skripsi ini serta saran – saran untuk pengembangan lebih lanjut.

# BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini berisi pembahasan tentang teori dasar yang berhubungan dengan perancangan dan pembangunan *game*.

## 2.1 Penelitian Terkait

Pada kurikulum pendidikan taman kanak-kanak (TK) terdapat mata pelajaran mengenai pengenalan bentuk geometri atau bangun datar sebagai dasar anak mengenal konsep dasar matematika sebelum memasuki usia sekolah.

Penelitian sebelumnya yang terkait dengan penulisan skripsi ini adalah penelitian yang berjudul "Perancangan Media Interaktif Dan *Game* Untuk Kompetensi Mengelompokkan Di Paud" ditulis oleh Muhammad Ta'yinul Karim (2011). Tujuan penelitian tersebut adalah dapat membantu penyampaian materi secara interaktif dengan *game* edukasi sekaligus memperkenalkan perangkat teknologi informasi dan komunikasi kepada siswa sehingga dapat mengejar ketertinggalan siswa tentang IPTEK dalam pendidikan. Kekurangan dalam jurnal penelitian ini adalah konsep rancangan yang mengacu pada media interaktif, sehingga untuk pembuatan *game* hanya dilakukan dengan pembuatan skenario bagaimana *game* berjalan, tanpa melibatkan konsep *game development*.

## 2.2 *Game*

Dalam kamus bahasa Indonesia *Game* adalah permainan. Permainan merupakan bagian dari bermain dan bermain juga bagian dari permainan keduanya saling berhubungan. *Game* adalah suatu aktifitas yang membutuhkan paling sedikit satu pemain, memiliki peraturan, serta memiliki kondisi kemenangan. (Pedersen, 2008)

### 2.1.1 *Element Game*

Ian Schereiber & John Sebastian (2009) mendefinisikan beberapa elemen dasar yang membentuk sebuah *game* yaitu *Player*, *Objective (goals)*, *Rules*, *Resources and Resources Managament*, *Game State*, *Information*, *Sequencing*, *Player Interaction*, *Theme* dan *Games as System* (Schereiber, 2009).

**1. *Player***

*Player* adalah peserta dalam permainan. Berapa banyak pemain yang dapat menjalankan permainan, berapa jumlah yang pasti atau tepat (4 pemain saja) atau sejumlah dalam variable (2 sampai 5 pemain)? ,dapatkah pemain memasuki atau meninggalkan permainan saat bermain?, bagaimana hal ini mempengaruhi permainan?, ada tim atau hanya individu? dan apa hubungan antara pemain?.

**2. *Objective (goals)***

*Objective*(*goals)* merupakan suatu tujuan yang dicapai dalam sebuah permainan serta hal yang pemain coba lakukan untuk mencapai tujuan tersebut.

**3. *Rules***

*Rules* adalah aturan main. Ada tiga kategori aturan dalam *game* yaitu *setup* (hal yang anda lakukan sekali diawal pertandingan), *progression of play* (apa yang terjadi selama pertandingan), dan *resolution* (kondisi apa yang menyebabkan permainan sampai akhir, dan bagaimana hasil yang ditentukan berdasarkan pada keadaan permainan).

**4. *Resources* dan *Resources Managament***

*Resources* dan *resources management* disini mencakup sumber daya eksplisit (jelas) pada game, jenis sumber daya yang dapat dikontrol oleh pemain dan sumber daya yang bisa di manipulasi saat bermain.

**5. *Game State***

G*ame state* untuk memberikan gambaran tentang suatu objek pada suatu titik waktu tertentu dalam sebuah *game*.

**6. *Information***

Informasi dalam *game* yang tersedia bagi pemain memiliki efek drastis pada permainan. Informasi tersebut ada yang mencakup hanya beberapa informasi yang bersifat pribadi bagi setiap individu seperti permainan poker, informasi total dimana semua pemain melihat keadaan permainan lengkap setiap saat seperti catur, informasi dimana hanya satu pemain dapat memiliki informasi rahasia mereka sendiri, sementara yang lain tidak, seperti *Scotland Yard*, dan informasi dalam permainan itu sendiri dapat berisi informasi yang tersembunyi dari semua pemain seperti *clue dan sleuth.*

**7. *Sequencing***

*Sequencing* dalam *game* adalah penentuan alur tindakan pada sebuah permainan. S*equencing* merupakan rangka apa yang pemain lakukan untuk mengambil tindakan, bagaimana alur permainan mengalir dari satu tindakan ke tindakan yang lain dan *game* dapat bekerja secara berbeda tergantung pada struktur gilirannya yang digunakan.

**8. *Player Interaction***

Interaksi pemain sering diabaikan dalam *game* tetapi sangat penting dalam permainan untuk dipertimbangkan. Interaksi pemain adalah bagaimana pemain berinteraksi satu sama lainnya dan bagaimana mereka dapat mempengaruhi satu sama lainnya.

**9. *Theme***

Tema dalam *game* merupakan bagian dari permainan yang tidak secara langsung mempengaruhi *gameplay*.

**10. *Game as System***

*Game as system* adalah kombinasi dari hal-hal atau bagian yang membentuk keseluruhan yang kompleks pada permainan.

### 2.1.2 *Genre Game*

*Genre* atau tipe *game* dapat dikategorikan menjadi *action*, *shooter*, *adventure*, *construction/management*, *life simulations*, *music/rhythm*, *party*, *puzzle*, *sports*, *strategy*, dan *vehicle simulation*. Tidak menutup kemungkinan ada *game* yang mengadopsi lebih dari satu *genre* (*hybrid*). Berikut adalah penjelasan mengenai beberapa kategori *genre game* tersebut (Pedersen, 2008).

a. *Action game*: *game* yang menuntut pemainnya terus melakukan gerakan seperti bergerak, menyerang, bergerak, bereaksi, lalu bergerak lagi pada karakter permainannya. Membutuhkan koordinasi mata dan tangan untuk memainkannya.
b. *Shooter game: game* yang pemainnya lebih memfokuskan menembakkan proyektile pada musuh, ketika bergerak dengan cepat dan berganti tampilan kamera.
c. *Adventure game: game* petualangan dimana teka teki muncul selama perjalanannya dengan mengumpulkan *item* dan memanajemen persediannya.
d. *Construction/Management game: game* yang pemainnya melakukan pembangunan dan memperluas wilayahnya dengan sumber daya yang terbatas.
e. *Life Simulations game: game* yang meniru realita di kehidupan sehari-hari.
f. *Music/Rhythm game: game* yang pemainnya bermain untuk mencocokan irama atau mengalahkan lawan dengan mendapatkan point tertinggi.
g. *Party game: game* yang dirancang khusus untuk beberapa pemain yang lebih banyak dan didasari dengan permainan yang kompetitif.
h. *Puzzle game: game* yang didasarkan pada logika dan penyelesaian pola. Mereka dapat dimulai dengan lambat, mempunyai metode, atau menggunakan tangan dan koordinasi mata.
i. *Sports games: game* dengan tema olahraga tradisional maupun *modern*. Saat ini dapat berupa POV (*point of view*) *game,* juga bisa berupa *strategy game* dimana pemain biasanya memiliki peran sebagai manajer.
j. *Strategy games: game* yang membutuhkan pemikiran dan perencanaan. Pemenang *game* ditentukan dengan "*battle of the minds*".
*k.* *Vehicle Simulation game: game* yang memberikan simulasi kepada pemain dalam mengendari kendaraan mulai dari mobil sampai pesawat luar angkasa dengan memberikan pengalaman senyata mungkin.

## 2.3 Bangun Datar

Bangun datar dapat didefinisikan sebagai bangun yang rata yang mempunyai dua dimensi yaitu panjang dan lebar tetapi tidak mempunyai tinggi dan tebal (Tarigan, 2006).

Bangun datar adalah bangun yang dibuat (dilukis) pada permukaan datar, contohnya bangun bersisi 4 disebut bangun datar karena seluruh bangun terletak dalam bidang yang datar (Negoro & Harahap, 2003).

Bangun datar adalah bangun yang mempunyai sisi dan sudut (Kusni, 2008), diantaranya:

1. Segitiga adalah bangun yang memiliki tiga sisi.
2. Jajar genjang adalah suatu segi empat yang sisi-sisinya sepasang sejajar.
3. Persegi panjang adalah suatu jajargenjang yang satu sudutnya siku-siku.
4. Belah ketupat adalah suatu jajar genjangyangdua sisinya berurutan sama panjang.
5. Trapesium adalah suatu segi empat yang memliliki tepat sepasang sisi yang sejajar.
6. Lingkaran adalah garis lengkung yang bertemu kedua ujungnya yang merupakan himpunan titik-titik yang berjarak sama dari sebuah itik tertentu.
7. Persegi adalah suatu segi empat yang empat sisinya berurutan sama panjang.

## 2.4 *Game Concept Design*

Tahap *game concept design* dilakukan dengan pembuatan *game design document*. Pada tahap pembuatan *game design document,* langkah yang pertama kali dilakukan adalah mencari ide. Pencarian ide dapat dilakukan dengan menuliskan ide yang ada atau berdiskusi dengan kelompok orang yang memiliki disiplin ilmu yang berbeda. Kemudian dari ide-ide yang didapatkan akan ditulis pada sebuah catatan yang disebut dengan *brainstorming note*. *Brainstroming note* disusun dalam sebuah *one sheet document* agar lebih terstruktur sehingga pembaca awam mengerti ringkasan tentang *game* yang dibuat.

Langkah selanjutnya adalah membuat *ten page design document*, isi dari dokumen ini adalah detail dari *one sheet document*. Dokumen ini maksimal terdiri dari 10 halaman dimana isi dari dokumen ini hanya bagian-bagian penting dari game yang dibuat seperti *title, gameplay, game controller, character, game interface*, dan lain-lain sehingga pembaca mengerti secara garis besar *game* yang dibuat tanpa harus mengetahui produk final *game*. Langkah terkahir dalam *game concept design* ini adalah membuat *game design document* yang merupakan detail *dari ten page design document*. Tujuan dibuatnya dokumen ini adalah untuk mengkomunikasikan kepada tim untuk meminimalisir kesalahan pada saat implementasi karena kurang mengerti tentang *game* yang dibuat. (Rogers, 2010)

## 2.5 MDA *Framework*

MDA *Framework* merupakan sebuah konsep cara men-*design* sebuah *game* bagi para *Game Designer.* Sebuah *game* didefinisikan dalam hal *Mechanics*, *Dynamics*, dan *Aesthetics*:

a. *Mechanics* merupakan sinonim untuk "*rule*" dalam sebuah *game*. Ini adalah batasan di mana *game* berjalan. Bagaimana mengatur *game*? Hal apa yang bisa pemain lakukan, dan apa yang akan terjadi bila pemain melakukan tindakan didalam *game state*? Kapan *game* berakhir dan bagaimana resolusi ditentukan? Semua ini didefinisikan oleh *mechanics*.
b. *Dynamics* menggambarkan bagaimana memainkan *game* ketika *rule* sudah ditetapkan. Strategi apa yang muncul dari *rule* tersebut? Bagaimana pemain berinteraksi satu sama lain?
c. *Aesthetics* adalah respon emosional dari pemain yang muncul ketika memainkan *game*. (Hunicke, et al., 2004)

## 2.6 Unity 3D

Unity 3D adalah sebuah aplikasi *game engine* yang memungkinkan penggunanya dapat membuat sebuah *game* 3D dengan mudah dan cepat. Secara *default*, Unity telah diatur untuk pembuatan *game* ber-*genre First Person Shooter* (FPS), namun Unity juga bisa digunakan untuk membuat *game* ber-*genre Role Playing Game* (RPG), dan *Real Team Strategy* (RTS). Selain itu Unity merupakan sebuah *engine multiplatform* yang memungkinkan *game* yang telah dibuat dapat digunakan untuk berbagai *platform* seperti Windows, Mac, Android, IOSS, PS3 dan juga Wii (Roedavan, 2014).

## 2.7 Bahasa Pemrograman C#

C# merupakan sebuah bahasa pemrograman yang sederhana, modern, berbasis objek, dan *type-safe*. C# berakar pada keluarga bahasa C dan programmer C, C++ dan Java dapat familiar dengan cepat. C# distandarisasi oleh ECMA International sebagai standar ECMA-334 dan oleh ISO/IEC sebagai standar ISO/IEC 23270. Microsoft C# Compiler untuk .NET *Framework* merupakan implementasi yang sesuai dari kedua standar tersebut (Hejlsberg, et al., 2011).

## 2.8 Pengujian Perangkat Lunak

Pengujian (*testing*) arsitektur dari perangkat lunak berorientasi objek menghasilkan sekumpulan *layered subsystems* yang mengenkapsulasi kelas-kelas yang berkolaborasi. Setiap elemen sistem (subsistem dan *class*) melakukan fungsi yang membantu untuk mencapai kebutuhan sistem. Hal ini sangat penting untuk menguji sebuah *object-oriented system* pada berbagai macam level yang berbeda dalam sebuah usaha untuk menemukan kesalahan-kesalahan yang mungkin terjadi dari kolaborasi kelas-kelas dan komunikasi subsistem melewati *architetural layer* (Pressman, 2001).

Pengujian perangkat lunak memerlukan perancangan kasus uji (*test case*) agar dapat menemukan kesalahan dalam waktu singkat dan usaha minimum. Berbagai macam metode perancangan kasus uji telah berevolusi. Metode ini menyediakan pendekatan sistematis untuk pengujian oleh *developer*. Terlebih lagi metode ini menyediakan mekanisme yang dapat membantu memastikan kelengkapan dari pengujian dan menyediakan kemungkinan tertinggi untuk menemukan kesalahan-kesalahan dalam perangkat lunak (Pressman, 2001). Teknik atau metode perancangan kasus uji yang digunakan adalah *black-box testing* dan *white-box testing*.

### 2.7.1 Blackbox Testing

*Black box testing* adalah teknik pengujian yang menguji hanya berdasarkan kebutuhan dan spesifikasi. *Black box testing* juga disebut sebagai *behavioral testing* dan berfokus pada kebutuhan fungsi dari perangkat lunak (Pressman, 2001). Proses umum yang terjadi pada *black box testing* yaitu:

a. Kebutuhan atau spesifikasi dianalisa terlebih dahulu.
b. Penentuan *input* valid terpilih berdasarkan spesifikasi untuk menentukan perangkat lunak berjalan dengan benar. *Input* yang tidak valid juga harus dipilih untuk memverifikasi bahwa perangkat lunak dapat mendeteksinya dan menanganinya dengan baik.
c. Penentuan *output* yang diharapkan sesuai dengan *input* yang telah dipilih.
d. Pengujian dibuat dengan *input* yang telah dipilih.
e. Pengujian dijalankan.
f. *Output* yang sebenarnya dibandingkan dengan *output* yang diharapkan
g. Penentuan dibuat menyangkut perangkat lunak berfungsi sesuai dengan spesifikasi yang telah ditentukan.

### 2.7.2 White-Box Testing

*White box testing* adalah teknik pengujian yang menguji berdasarkan jalur internal, struktur dan implementasi dari perangkat lunak yang sedang diuji. *White box testing* adalah teknik pengujian yang menggunakan struktur kontrol dari prosedur yang terdapat dalam perancangan untuk membuat kasus uji (Pressman, 2001). Ada dua jenis pengujian yang termasuk *white box testing* yaitu *basis path testing* dan *control structure testing.*

*Basis path testing* adalah teknik dalam metode *white box testing* untuk menyediakan desain *test case* yang dapat digunakan untuk menghitung kompleksitas desain prosedural untuk menghitung banyaknya jalur eksekusi. *Test case* tersebut digunakan untuk menjamin apakah tiap statement telah dijalankan minimal satu kali tiap pengujian. Dalam konteks *basis path*, nilai yang dihitung di dalam *cyclomatic complexity* menjelaskan banyaknya jumlah jalur independent yang harus disediakan dalam pengujian *basis path* (Pressman, 2001).

## 2.9 *Play Testing*

*Play testing* adalah suatu cara menguji sebuah *game* dengan cara memainkan *game* tersebut dan mengamati hal-hal apa saja yang muncul ketika *game* dimainkan. Tujuan dari *play testing* umumnya adalah untuk membuat *game* menjadi lebih baik.

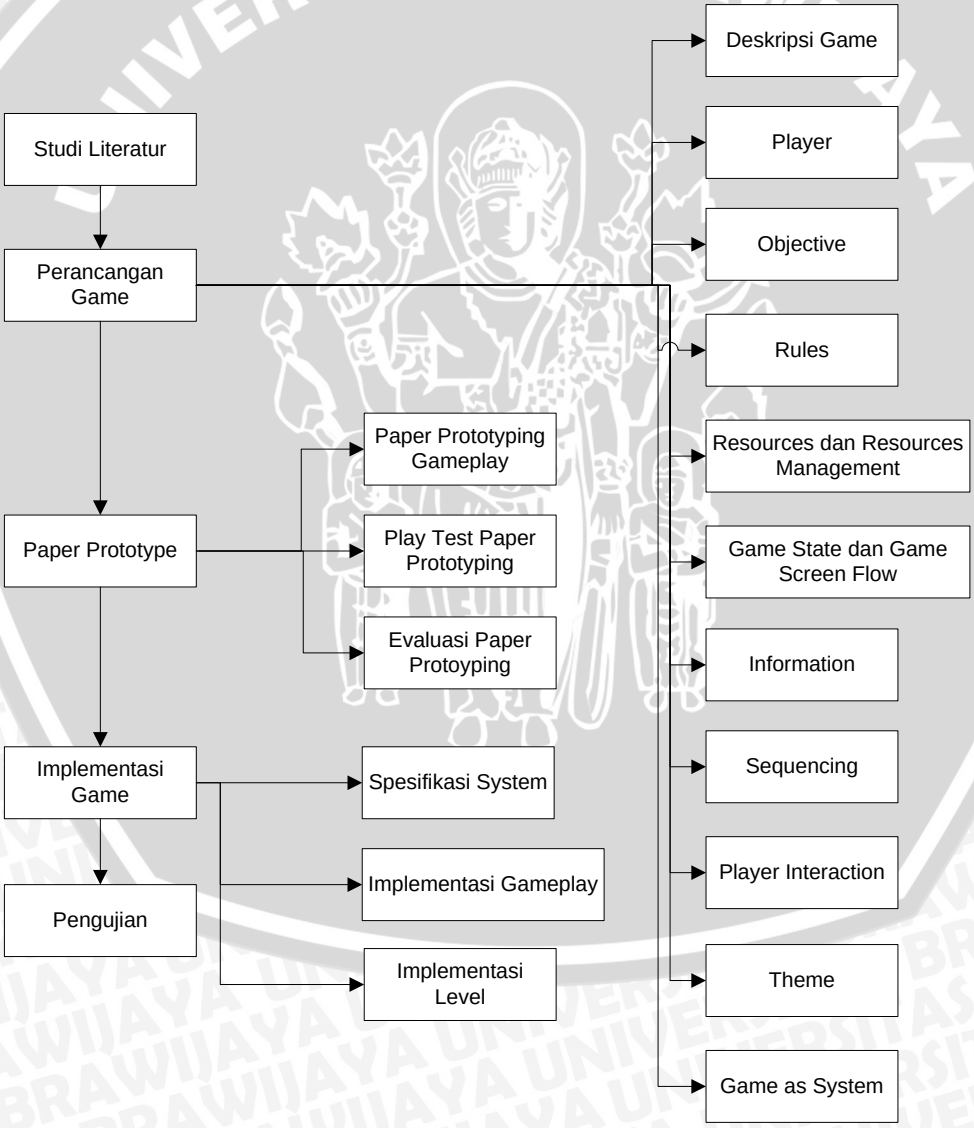
*Focus testing* merupakan salah satu dari sekian banyak jenis *play testing*. *Focus testing* adalah dimana proses *game* yang dibuat ini diuji kepada para *user* untuk melihat respon mereka terhadap *game* yang dibuat dengan mengajukan pertanyaan terkait. Sebagai contohnya apakah *game* yang telah dibuat susah atau mudah untuk dimainkan? Apakah *UI* (*User Interface*) pada *game* mudah digunakan? Apakah *game* menyenangkan setelah dimainkan?

# BAB 3 METODOLOGI

Pada bab ini akan dijelaskan langkah-langkah metodologi yang akan dilakukan dalam penelitian dan perancangan *game*. Pada metode penelitian akan dijabarkan mengenai langkah-langkah yang digunakan dalam penelitian skripsi dan dalam subbab perancangan menjelaskan perancangan *game* dari penelitian ini.

## 3.1 Metode Penelitian

Pada metode penelitian langkah-langkah yang dilakukan dimulai dengan studi literatur, perancangan *game*, p*rototype*, implementasi serta hasil dan pembahasan. Diagram alir metode penelitian ditunjukkan pada Gambar 3.1.




**Gambar 3. 1 Blok Diagram Runtutan Pengerjaan Penelitian**

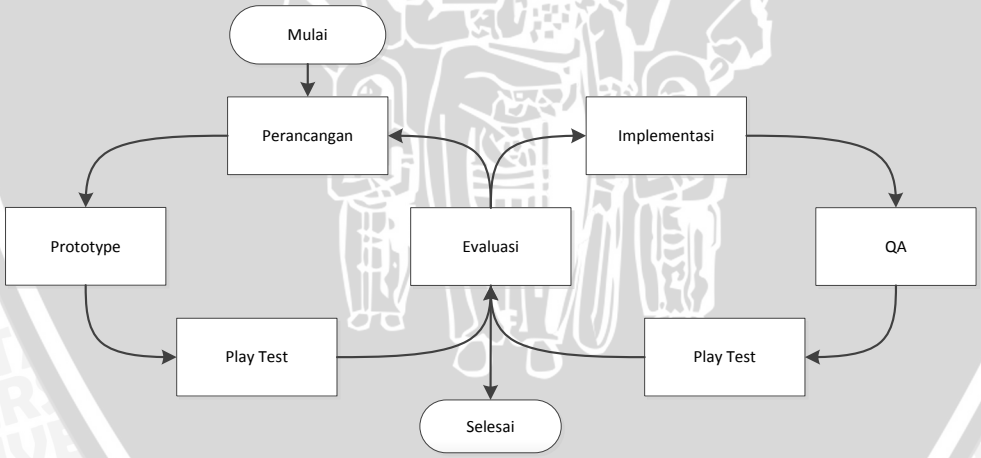
### 3.1.1 Studi Literatur

Studi literatur merupakan penelitian terkait yang membahas penelitian serupa sebelumnya dan landasan dasar teori yang akan digunakan untuk mendukung penulisan skripsi ini. Landasan teori tersebut antara lain :

1. Penelitian terkait
2. *Game*
3. Bangun Datar
4. *Game Concept Design*
5. MDA *Framework*
6. Unity 3D
7. Bahasa Pemrograman C#
8. Pengujian Perangkat Lunak
9. *Play Testing & Focus Testing*

### 3.1.2 Perancangan *Game*

Perancangan *game* atau *game design* adalah proses merancang atau men-*design* sebuah *game* yang bertujuan untuk memberi gambaran secara umum kepada pengguna tentang konsep *game* yang akan dibuat. Pada bagian perancangan ini menggunakan metodologi *iterative* dengan *rapid prototyping* yang ditunjukan pada Gambar 3.2.




**Gambar 3. 2 Metodologi *Game Design***
**Sumber :** (Schereiber, 2009)

*Iterative* dengan *rapid prototyping* adalah sebuah metode pengembangan *video game* dengan menggunakan konsep MDA *framework* yang dimulai dengan perancangan untuk menentukan *elemen* formal dari *game* yang akan dibuat, membuat *prototype game* berdasarkan hasil dari perancangan agar bisa dimainkan, melakukan *play testing* dengan prototype untuk mengevaluasi apakah *rule* yang dibuat sudah sesuai tujuan dan menyenangkan, apabila belum maka akan kembali ke perancangan untuk menentukan *rule* kembali, membuat *prototype*, *play testing*, dan evaluasi. Begitu seterusnya sampai *rule* yang kita buat sudah sesuai dengan tujuan, jika sudah sesuai maka akan berlanjut ke tahap selanjutnya yaitu implementasi.

Implementasi yang dilakukan adalah membuat rancangan yang sudah sesuai dengan tujuan ke dalam bentuk *digital* dan menjadikannya menjadi sebuah *video game.* Kemudian *video game* akan di mainkan untuk mengevaluasi apakah ada *error* atau *bug*. Jika ada maka akan dilakukan implementasi ulang, jika tidak ada maka *video game* telah selesai dibuat.

### 3.1.3 *Paper Prototype*

Setelah melakukan perancangan *game* dimana untuk menentukan *rule* atau *mechanic* dalam *game*, kemudian akan dilanjutkan dengan *paper prototype. P*embuatan *paper prototyping* akan mempermudah penulis dalam mengimplementasikan *game* dari hasil perancangan dan juga untuk mengetahui *dynamic* dan *astethic* yang dihasilkan dari *mechanic* yang dibuat. *Paper prototype* di bagi menjadi 3 tahap yaitu pembuatan *paper prototype gameplay*, *play test paper prototyping* dan evaluasi *paper prototyping*.

### 3.1.4 Implementasi *Game*

Implementasi *game* ini merupakan implementasi dari hasil evaluasi *paper prototyping* menjadi sebuah *game digital.* Pada implementasi ini menggunakan aplikasi Adobe Photoshop CS5 dan Adobe Ilustrator CS5 untuk pembuatan *asset* 2D, sedangkan pembangunan perangkat lunak *game* menggunakan *game engine* Unity versi 4.6 dengan bahasa pemrograman C#.

Implementasi *game* dibagi menjadi 3 tahap yaitu spesifikasi *system*. implementasi *gameplay* dan implementasi *level*.

### 3.1.5 Pengujian

Pengujian dan pembahasan pada *game* ini bertujuan untuk menemukan kesalahan dan kekurangan pada *game* tersebut. Metode pengujian yang digunakan adalah *white box* dengan teknik pengujian *basis path* dan pengujian *black box* dengan 2 teknik yaitu *tes flow diagram* dan pengujian terhadap pengguna *(focus testing).* Pengujian ini bertujuan untuk mengetahui *game* yang dibuat apakah sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut.

## 3.2 Perancangan Game

Perancangan *game* atau *game design* adalah proses merancang atau men-*design* sebuah *game* yang bertujuan untuk memberi gambaran secara umum kepada pengguna tentang konsep *game* yang akan dibuat dan untuk menentukan *mechanic* yang akan bekerja pada *game*.

Pada tahap perancangan *game* ini akan dijelaskan tahapan dalam pembuatan *game* yang akan dimulai dengan menjelaskan deskripsi *game* yang akan dibangun dan *element-element* formal yang ada di *game* yaitu *player*, *objective (goals), rules*, *resource & resource management*, *game state*, *information*, *sequencing*, *player interaction* dan terakhir *theme game* itu sendiri.

### 1. Deskripsi *Game*

*Game* "Shape Adventure" adalah *game* bergenre *puzzle match* dengan konten edukasi yang mengenalkan tentang bangun datar 2D. Konsep utama dari *game* "Shape Adventure" ini ditunjukan pada Tabel 3.1.

| No | Elemen | Keterangan |
|---|---|---|
| 1 | Judul *game* | Shape Adventure |
| 2 | *Game platform* | PC |
| 3 | Target *device* | PC, Notebook |
| 4 | Target usia | 4-6 tahun |
| 5 | *Genre game* | *Adventure Puzzle – Education* |

**Tabel 3. 1 *Main Game Concepts (lanjutan)***

| | | |
|---|---|---|
| 6 | USP (*Unique Selling Point*) | 1. Sebagai multimedia pembelajaran dalam bentuk *game*.<br>2. Sebagai media alternatif pembelajaran mengenal bangun datar 2D.<br>3. *Design UI & art 2D* yang menarik dan disesuikan dengan tema anak-anak. |

### 2. *Player*

*Player* adalah seseorang yang memainkan *game.* Dalam *game* ini hanya bisa dimainan oleh 1 *player*. Pemain disini bertugas untuk mengumpulkan beberapa bangun datar 2D dan menyelesaikan *puzzle* bangun datar 2D.

Lalu identifikasi aktor dilakukan untuk menjelaskan interaksi yang dilakukan aktor pada *game* ini yang ditunjukan pada Tabel 3.2.

**Tabel 3. 2 Identifikasi Aktor**

| No | Aktor | Deksripsi |
|---|---|---|
| 1 | *Player* | Pemain mengontrol *character* untuk mencari bangun datar 2D. Bangun datar 2D ini nantinya akan dicocokkan pada gambar yang telah disediakan. |

**3. *Objective (goals)***

*Game* goal pada game adalah mengumpulan beberapa bangun datar 2D dan mencocokkannya pada gambar yang sudah di sediakan.

**4. *Rules***

Berikut dijelaskan *rule* pada *gameplay* dalam *game* ini, yaitu:

- ***Collecting Item***

Pemain akan menggerakan satu *character* untuk mencari *item* yang berbentuk bangun 2D dengan melewati beberapa halangan dan rintangan. Apabila *character* yang dimainan jatuh atau terkena jebakan maka *player* akan mengulangi dari awal. Setelah *player* mengumpulkan semua item maka *player* harus melaju ke garis *finish*.

- ***Item Match***

Pada *state* ini *player* akan mencocokkan *item* yang didapat dengan gambar yang telah disediakan. Gambar tersebut memiliki bagian yang hilang, dan untuk menampilkan bagian yang hilang tersebut *player* harus mencocokan *item* yang di dapat dengan bagian tersebut. Namun *player* hanya memiliki waktu 1 menit untuk menyelesaikan permainan tersebut apabila sampai waktu yang ditentukan *player* tidak bisa menyelesaikan maka akan *game over* dan tidak akan mendapatkan bintang.

**5. Resources *dan Resources Management***

Jenis sumber daya yang dapat dicontrol oleh pemain dan sumber daya yang bisa di manipulasi saat bermain yaitu:

- *Item*
- *Time*

### 6. Game *State* & *Game Screen Flow*

Pada tahap ini akan dijelaskan rancangan *game state* dan *game screen flow* sebagai berikut :

- ***Game State***

*Game state* disini untuk memberikan gambaran tentang suatu objek pada suatu titik waktu tertentu dalam sebuah *game* yang ditunjukan pada tabel. Terdapat 2 *game state* berbeda didalam *game* ini yang ditunjukan pada Tabel 3.3.

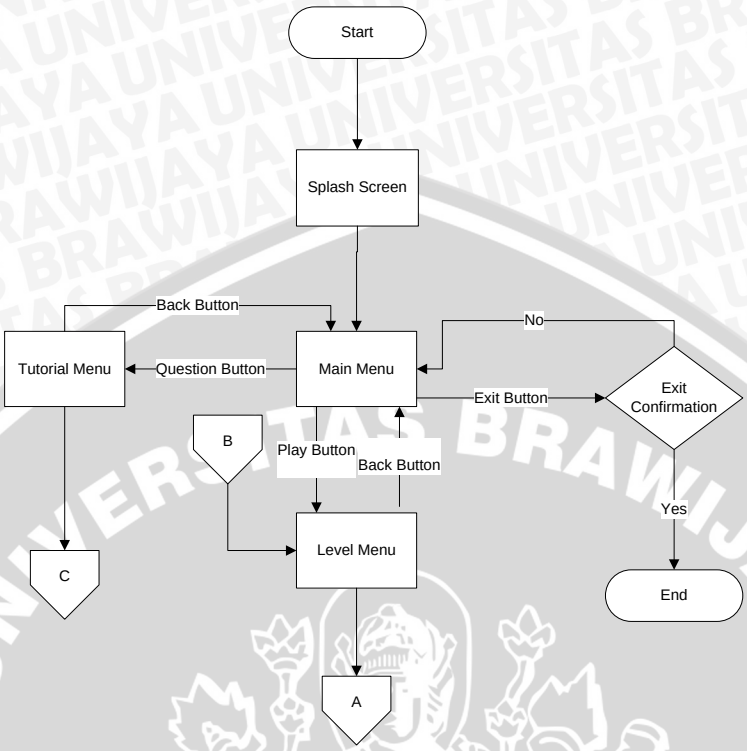
**Tabel 3. 3 *Game State***

| No. | *Game State* | Keterangan |
|---|---|---|
| **1** | *Game state collecting item* | Merupakan *state* dimana *player* mencari *item* dengan menjalankan *character* melewati halangan dan rintangan. |
| **2** | *Game* state *item match* | Merupakan *state* dimana pemain akan mencocokkan *item* yang didapat dengan gambar yang memiliki bagian-bagian yang hilang. |

- ***Game Screen Flow***

*Game screen flow* digunakan untuk menjelaskan urutan dari aktivitas dalam *game* ini. *Game screen flow* menggambarkan alur kerja mulai dan memberikan detail alur percabangan yang terdapat pada proses suatu *event* dalam aktivitas dari *starting point* hingga *finish point.* Berikut adalah *game screen flow* dari *game* ini :

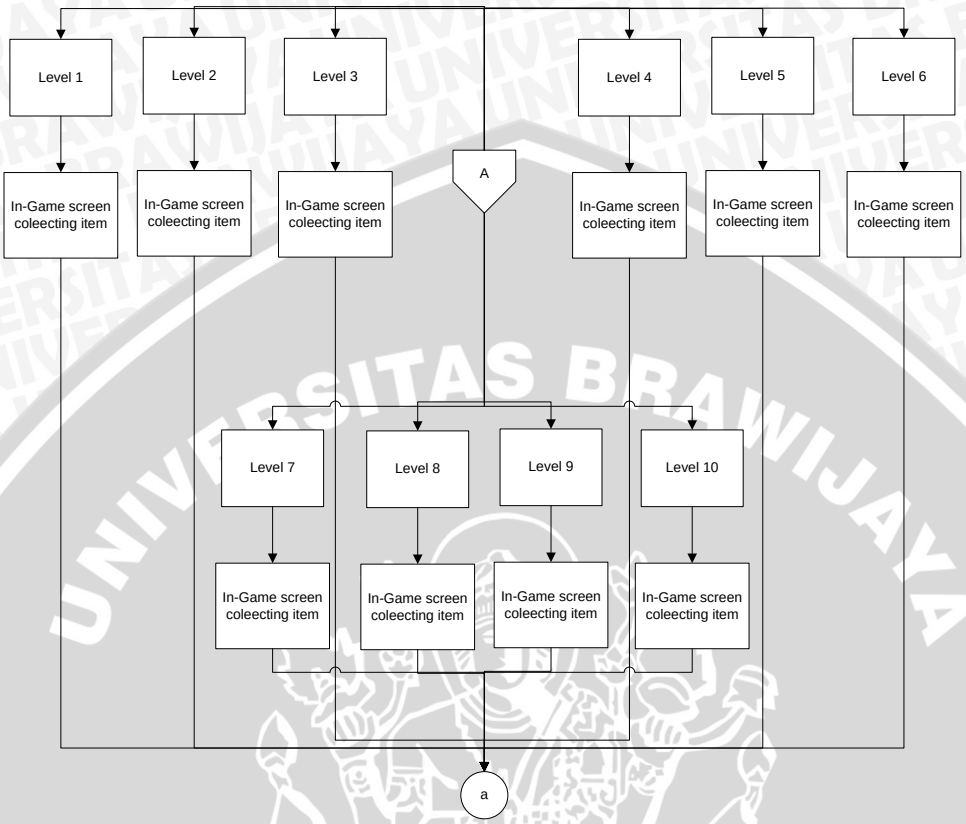
**a. *Game Screen Flow Main Menu***




**Gambar 3. 3 *Game Screen Flow* Menu**

Saat pertama kali menjalankan *game*, pemain akan melihat *splash screen* lalu masuk ke dalam *main menu* dari *game*. Dari *main menu* terdapat 3 pilihan yang dapat diakses oleh pemain. Pilihan *tutorial menu* akan menampilkan bagaimana cara memainkan *game*. Pilihan *level menu* akan menampilkan jumlah *level* yang dapat dipilih dan dimainkan pemain. Pilihan *exit* untuk membuat pemain keluar dari *video game*.

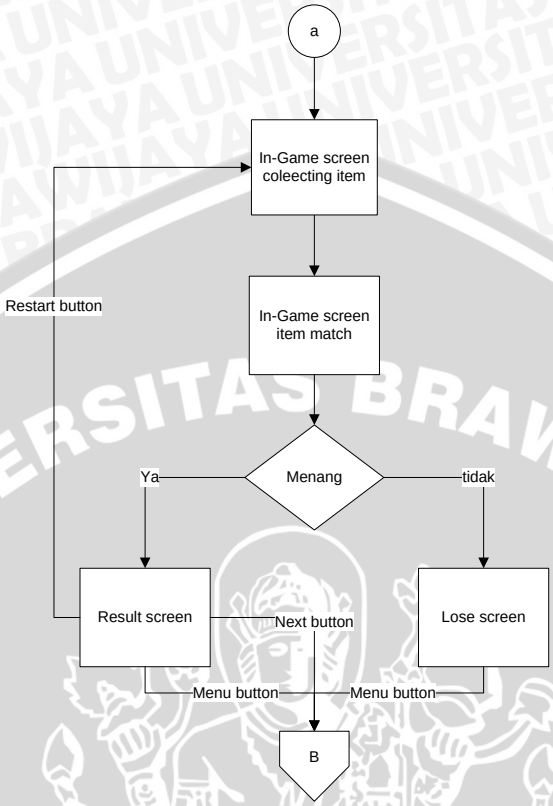
**b. *Game Screen Flow Level Select***




**Gambar 3. 4 *Game Screen Flow Level Select***

Saat pemain memasuki *level menu*, akan terdapat pilihan 10 *level* yang disediakan. Setiap *level* yang dipilih akan memulai permainan dengan *gameplay collecting item*. Namun saat pertama kali *game* dimulai, pilihan yang disediakan hanya 1 yaitu *level* 1 sedangkan 9 *level* yang lain masih di kunci dan tidak bisa dipilih. Untuk pilihan *level* yang lain akan otomatis terbuka jika pemain sudah menyelesaikan *level* sebelumnya. Pemain harus menyelesaikan permainan dengan *gameplay collecting item* untuk melanjutkan ke tahap selanjutnya yaitu permainan dengan *gameplay item match*.

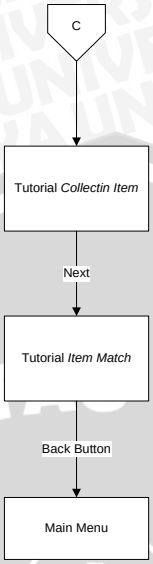
### c. *Game Screen Flow In Game Item Match*




**Gambar 3. 5 *Game Screen Flow In Game Item Match***

Didalam *gameplay item match* pemain harus menyelesaikan permainan untuk dapat membuka *level* selanjutnya. Jika pemain dapat memenangkan permainan maka akan menampilkan *result screen* dan pilihan *menu next, restart*, serta *main menu.* Namun jika kalah akan menampilkan *lose screen* dan pilihan *menu restart* serta *main menu.*

**d. *Game Screen Flow Tutorial***




**Gambar 3. 6 *Game Screen Flow Tutorial***

Saat pemain berada pada halaman *tutorial menu*, pemain mendapatkan informasi bagaimana cara memainkan *game*. Dimulai dengan informasi bagaimana cara memainkan *game* saat berada pada *gameplay collecting item.* Saat pemain menekan tombol *next* maka pemain akan mendapatkan informasi bagaimana cara memainkan *game* saat berada pada *gameplay item match*. Untuk kembali ke *main menu* pemain cukup menekan tombol *back*.

**7. *Information***

Informasi yang ada pada *game* ini hanya untuk menginformasikan bagaimana cara untuk memainkan *game* ini berupa *tutorial* permainan pada *gameplay collecting item* dan *item match*.

**8. *Sequencing***

Tidak ada perancangan *sequencing* didalam *game* ini dikarenakan *game* ini hanya dapat dimainkan oleh satu orang pemain saja.

**9. *Player Interaction***

Tidak ada *player interaction* didalam *game* ini dikarenakan *game* ini hanya dapat dimainkan oleh satu orang pemain saja.

**10. *Theme***

Berikut rancangan tema yang akan dibuat pada *game* ini yang ditunjukan pada Tabel 3.4.

**Tabel 3. 4 Perancangan *Theme***

| No | *Stage* | Tema | Keterangan |
|---|---|---|---|
| 1 | *Main Menu Game* | *Flat* | |
| 2 | *Level Select* | *Flat* | |
| 3 | *Tutorial* | *Flat* | |
| 4 | *Gameplay Colecting Item* | *Flat* | |
| | | Musik | Roald Strauss |
| 5 | *Gameplay Item Match* | *Flat* | |
| | | Musik | Roald Strauss |

Pada tabel diatas tema yang digunakan adalah *flat*. *Flat* merupakan gaya *design* yang menghilangkan unsur 3D seperti bayangan, gradien, tekstur dan lain sebagainya.

**11. *Game as System***

Di dalam *game* ini tidak terdapat fitur dan sistem yang dapat dipadukan menjadi sebuah sistem yang kompleks.

## 3.3 *Paper Prototype*

Setelah melakukan *brainstorming* pada perancangan dan menetapkan *element* formal termasuk *mechanic* yang akan digunakan dalam membuat *game*, selanjutnya adalah tahap dalam pembuatan *paper prototype*. *Paper prototyping* bertujuan agar dapat mengevaluasi dan mendapatkan *feedback gameplay* secara cepat, *stakeholder* (dalam hal ini perancang) dapat melihat, menyentuh, berinteraksi dengan *prototype* agar dapat memunculkan ide-ide secara visual dan mengembangkannya. Pada *paper prototyping* ini kita bisa mendapatkan *dynamic* dan *astethic game* dari *mechanic* yang sudah dibuat sebelumnya.

### 3.3.1 *Paper Prototype Gameplay*

Pada tahap pembuatan *paper prototype gameplay* ini dibagi menjadi dua *paper prototype gameplay* , yaitu *paper prototype* pada *gameplay collecting item* dan *item match*.

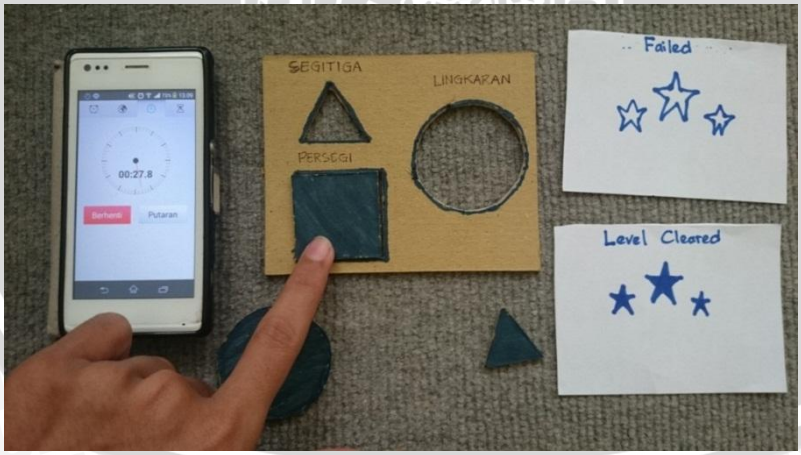
### 3.3.1.1 *Paper Prototype Gameplay Collecting Item*

Berikut *screenshot prototype gameplay collecting item* yang ditunjukan pada Gambar 4.1. Pada gambar terlihat sebuah dadu, 2 *coin*, *item,* pion berwarna kuning sebagai *character*.

**Gambar 3. 7 *Prototype Gameplay Collecting Item***

### 3.3.1.2 *Paper Prototype Gameplay Item Match*

Berikut *screenshot paper prototype gameplay item match* yang ditunjukan pada Gambar 3.8. Pada gambar terlihat bagian-bagian gambar *item*, *timer*, halaman *win* dan halaman *lose*.




**Gambar 3. 8 *Prototype Gameplay Item Match***

### 3.3.2 *Play Test Paper Prototyping*

Pada tahap ini akan dijelaskan tentang *play testing* dengan *paper prototype* yang akan dilakukan oleh *game designer*, yaitu *play test* pada *paper prototyping gameplay collecting item* dan *item match*. Tujuan dari *play test* ini untuk memperagakan *gameplay game* sebelum dibuat menjadi sebuah aplikasi *game*.

#### 3.3.2.1 *Play Testing Paper Gameplay Collecting Item*

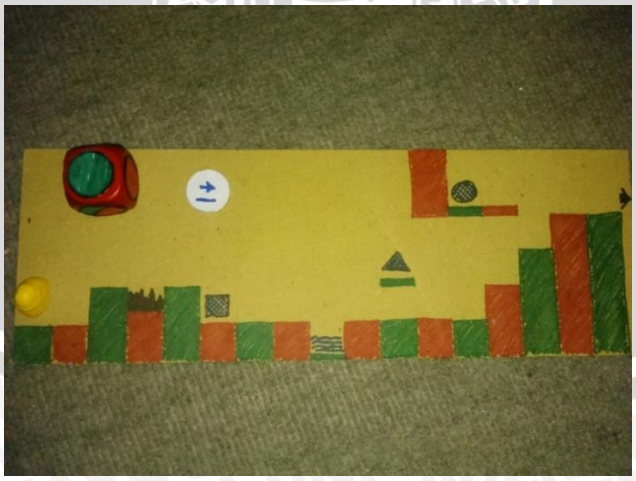
Pada tahap ini akan mencoba *play testing collecting item* dengan rancangan *rules & goals* yang ditunjukan pada tabel 3.5.

**Table 3. 5 Rancangan *Rules* & *Goals Paper Prototyping Collecting Item***

| No | Elemen | Keterangan |
|---|---|---|
| **1** | *Objective* | Mengumpulkan semua *item* yang tersedia. Dan pergi menuju titik *finish.* |
| **2** | *Setup* | Pemain akan mendapatkan sebuah dadu untuk menentukan langkah, 2 koin untuk menentukan arah, dan sebuah pion berwarna kuning sebagai *character*. |
| **3** | *Progression* | Pemain akan menggerakkan sebuah *character* untuk mencari *item*. |
| **4** | *Hazards* | *Spike traps* (sebuah jebakan berduri yang ada di tanah) |

Setelah itu akan dijelaskan bagaimana cara untuk memainkannya dengan contoh cara untuk menggerakan *character*, kondisi jika *character* terkena *hazard, character* berada pada gambar bangun 2D.

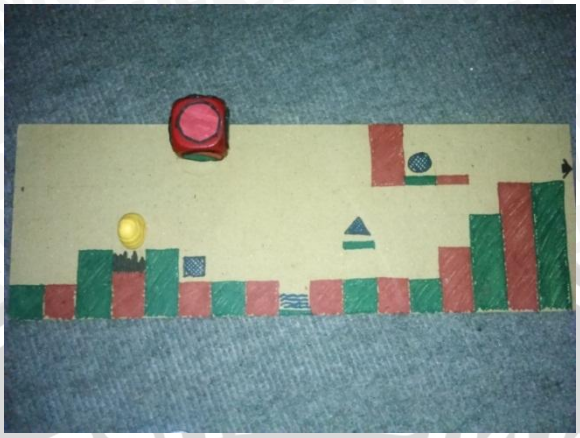
**1. Menggerakan *Character***



**Gambar 3. 9 *Play Test* Menggerakkan *Character***

Pada *play test* ini mensimulasikan bagaimana cara untuk menggerakan *character* menggunakan dadu untuk berjalan, dan koin untuk menentukan arah serta posisi awal pion *character* yang ditunjukkan pada Gambar 3.9.

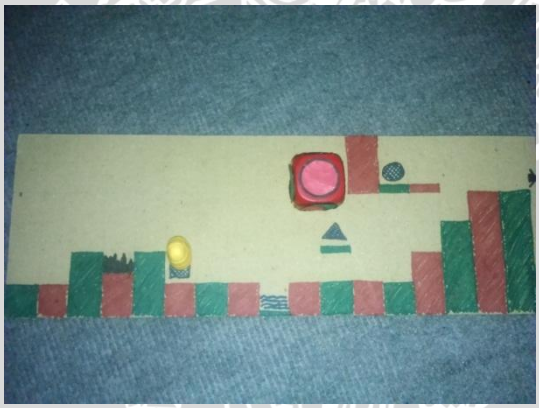
**2. Kondisi *Character* ketika terkena *hazard***



**Gambar 3. 10 *Play Test Character* Terkena *Hazard***

Pada *play test* ini mensimulasikan kondisi jika *character* terkena *hazard* di tampilkan pada Gambar 3.10**.** Pada Gambar 3.10 terlihat pion berada pada gambar *spike trap.* Apabila *character* berada pada kondisi ini maka posisi *character* akan berada pada titik awal sebelum permainan dimulai.

**3. Kondisi *Character* berada pada gambar bangun 2D**



**Gambar 3. 11 *Play Test Character* Berada Pada Gambar 2D**

**Gambar 3. 12 *Play Test Character* Mendapatkan *Item***

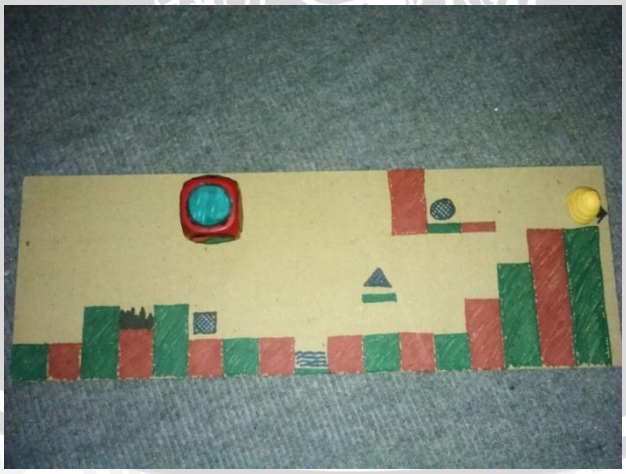
Pada *play test* ini mensimulasikan kondisi *character* ketika berada pada gambar bangun 2D seperti pada Gambar 3.11**.** Jika *character* berada pada posisi tersebut maka *player* akan mendapatkan item bangun 2D seperti pada Gambar 3.12.

**4. Kondisi *Character* melompat**

**Gambar 3. 13 *Play Test Character* Melompat**

Pada *play test* ini mensimulasikan kondisi *character* jika ingin melompat. *Player* harus mendapatkan warna hijau di dadu dan mendapatkan koin yang bertanda arah ke atas agar pion dapat melompat keatas seperti yang ditunjukan pada Gambar 3.13.

**5. Kondisi beralih ke gameplay item match**



**Gambar 3. 14 *Play Test* Beralih Ke *Gameplay Item Match***

Pada *play test* ini mensimulasikan kondisi dimana *player* akan berpindah ke *gameplay item match* apabila pion *character* berada di ujung dan menyentuh tanda panah ke kanan seperti yang ditunjukan pada Gambar 3.14.

#### 3.3.2.2 *Play Testing Paper Gameplay Item Match*

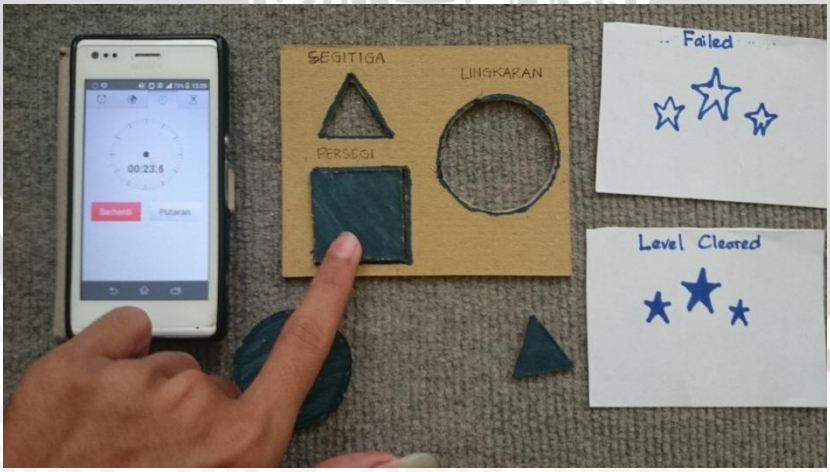
Pada tahap ini akan menggunakan *play testing item matching* dengan rancangan *rules & goals* yang ditunjukan pada Tabel 3.6.

**Table 3.6 Rancangan *Rules* & *Goals Paper Prototyping Gameplay Item Match***

| No | Elemen | Keterangan |
|---|---|---|
| 1 | Item | Item-item yang didapatkan ketika berada di *gameplay collecting item*. |
| **2** | *Progression* | Pemain akan mengisi bagian gambar yang kosong dengan *item* yang telah didapatkan pada *gameplay* sebelumnya. |
| **3** | *Time & life* | 30 detik. |
| **4** | *Reward & economy* | • Level pada *game* ini akan terbuka.<br>• Jika bisa menyelesaikan *puzzle* sebelum detik 20 maka akan mendapatkan *3 stars*<br>• Jika bisa menyelesaikan *puzzle* sebelum detik 10 maka akan mendapatkan *2 stars*<br>• Jika bisa menyelesaikan *puzzle* sebelum detik 0 maka akan mendapatkan *1 stars* |

Setelah itu akan dijelaskan bagaimana cara untuk memainkannya dengan contoh cara untuk melengkapi gambar dengan *item* yang sudah di dapat dan kondisi jika pemain menang dan kalah.

**1. Melengkapi bagian yang kosong**




**Gambar 3. 15 *Play Test Item Match* Menggerakan *Item***

Pada *play test* ini mensimulasikan bagaimana cara untuk menggerakan *item* bangun datar menuju kotak *puzzle* yang disediakan Gambar 3.15.

**2. Kondisi Menang**


**Gambar 3. 16 *Play Test Item Match* Kondisi Menang**

Pada *play test* ini mensimulasikan kondisi *game* ketika pemain sudah mengisi semua bagian yang kosong dengan *item* yang didapat seperti pada Gambar 3.16.

**3. Kondisi Kalah**


**Gambar 3. 17 *Play Test Item Match* Kondisi Kalah**

Pada *play test* ini mensimulasikan kondisi *game* ketika pemain telah kehabisan waktu dan kalah yang ditunjukan pada Gambar 3.17.

### 3.3.3 *Evaluasi Paper Prototyping*

Pada tahap ini evaluasi dilakukan dengan tujuan untuk menilai apakah hasil dari *play test* berdasarkan pada kriteria dan tujuan yang telah ditetapkan dan melihat *dynamic* serta *astethic* yang dihasilkan dari *mechanic game* yang dibuat, selanjutnya diikuti dengan pengambilan keputusan atas objek yang dievaluasi.

Evaluasi dilakukan dengan cara memainkan *paper prototyping* yang telah dibuat bersama para *play tester*. Hasil dari *mechanic game* berupa *dynamic* dapat diketahui disini dengan melihat apa saja cara yang dilakukan para *play tester* untuk menyelesaikan *game*. Sedangkan *astethic* dapat diketahui dengan melihat reaksi *play tester* setelah selesai memainkan *game*. Beberapa pertanyaan diberikan kepada para *play tester* setelah selesai memainkan *game*.

Berikut adalah beberapa hasil evaluasi yang dilakuan dengan menggunakan metode iterasi. Evaluasi yang pertama dilakukan oleh 5 orang *game designer* dari FILKOM-UB. Berikut hasil yang ditunjukan pada Table 3.7 dan Table 3.8

**Table 3.7 Hasil *Play Test Paper Prototyping* Pertama**

| No | Kriteria dan Tujuan | Yang Diharapkan | Nadia | Rusli | Andryanto | Abdi | Dwi |
|---|---|---|---|---|---|---|---|
| 1 | Apakah permainan ini menyenangkan? | Baik | Ya | Ya | Cukup | Ya | Cukup |
| 2 | Apakah permainan ini menarik? | Baik | Ya | Ya | Ya | Cukup | Cukup |
| 3 | Apakah prosedur dan *rule* permainan ini mudah untuk di mengerti? | Mudah | Mudah | Mudah | Mudah | Mudah | Mudah |
| 4 | Apakah *gameplay* permainan ini terlalu panjang, pendek atau tepat? | Tepat | Tepat | pendek | Tepat | Tepat | Tepat |
| 5 | Apakah jenis *gameplay* ini dapat membuat pemain lebih mengingat dan memahami bangun datar 2D? | Ya | Ya | Ya | Ya | Ya | Ya |

**Table 3.8 Kritik dan Saran *Play Test Paper Prototyping* Pertama**

| No | Pemain | Kritik & Saran |
|---|---|---|
| 1 | Nadia | Diberi suara nama bangun datar untuk *gameplay item match* ketika di implementasikan |
| 2 | Rusli | Bentuk *puzzle* lebih beragam |
| 3 | Andryanto | Kasih *iventori box* untuk *gameplay collecting item* ketika di implementasikan |
| 4 | Abdi | Tidak ada |
| 5 | Dwi | Waktu yang diberikan terlalu pendek jika di implementasikan dalam bentuk digital. |

Dari hasil evaluasi yang pertama terhadap para *game designer* dari FILKOM-UB, didapatkan hasil yang cukup memuaskan. Dari beberapa pertanyaan yang diajukan didapatkan respon yang positif seperti yang ditunjukkan Tabel 3.7. Table 3.8 merupakan saran yang didapatkan dari para *game designer* setelah memainkan *paper prototyping*. Dari beberapa saran yang didapatkan, umumnya para *game designer* memberikan masukan yang mengarah ke implementasi *game* dalam bentuk *digital*. Sedangkan saran yang mempengaruhi *design paper prototype* diberikan oleh Rusli, yaitu dengan menambah bentuk *puzzle* yang beragam.

Berdasarkan hasil dan saran yang didapatkan dari evaluasi yang pertama maka perlu diadakan perubahan dalam *paper prototype*. Perubahan yang dilakukan berupa mengganti waktu dalam *game play item match* menjadi 60 detik dengan sistem *reward* mendapatkan 3 *stars* apabila berhasil menyelesaikan *puzzle* sebelum waktu menunjukkan angka 40, mendapatkan 2 *stars* apabila berhasil menyelesaikan *puzzle* sebelum waktu menunjukkan angka 20 dan mendapatkan 1 *stars* apabila berhasil menyelesaikan *puzzle* sebelum waktu menunjukkan angka 0. Selain hal tersebut ditambah juga 1 level untuk memberikan tingkat kesulitan yang lebih tinggi dan gambar *puzzle* yang berbeda.

Kemudian dilakukan iterasi dengan merubah *paper prototype* sesuai dengan saran dan evaluasi yang didapatkan dan dilakukan evaluasi lagi terhadap *paper prototype* yang ke dua.

Evaluasi kedua dilakukan oleh 5 anak dari TK RA Muslimat NU 16 sebagai *play tester*. Berikut hasil yang ditunjukan pada Tabel 3.9 dan Tabel 3.10.

**Table 3.9 Hasil *Play Test Paper Prototyping* Kedua**

| No | Kriteria dan Tujuan | Yang Diharapkan | Anak ke-1 | Anak ke-2 | Anak ke-3 | Anak ke-4 | Anak ke-5 |
|---|---|---|---|---|---|---|---|
| 1 | Apakah permainan ini menyenangkan? | Baik | Ya | Ya | Ya | Cukup | Ya |
| 2 | Apakah permainan ini menarik? | Baik | Ya | Ya | Ya | Kurang | Ya |
| 3 | Apakah prosedur dan *rule* permainan ini mudah untuk di mengerti? | Mudah | Mudah | Mudah | Mudah | Mudah | Mudah |
| 4 | Apakah *gameplay* permainan ini terlalu panjang, pendek atau tepat? | Tepat | Tepat | Tepat | Tepat | Pendek | Tepat |
| 5 | Apakah jenis *gameplay* ini dapat membuat pemain lebih mengingat dan memahami bangun datar 2D? | Ya | Ya | Ya | Ya | Ya | Ya |

**Table 3. 10 Kritik dan Saran *Play Test Paper Prototyping* Kedua**

| No | Pemain | Kritik & Saran |
|---|---|---|
| 1 | Anak ke-1 | Gambar *puzzle* lebih beragam. |
| 2 | Anak ke-2 | Bentuk bangun datar 2D lebih beragam |
| 3 | Anak ke-3 | Tidak ada. |
| 4 | Anak ke-4 | Gambar *puzzle* lebih beragam |
| 5 | Anak ke-5 | Tidak ada |

Dari hasil evaluasi yang kedua terhadap para *play tester*, didapatkan hasil yang memuaskan. Dari beberapa pertanyaan yang diajukan didapatkan respon yang positif seperti yang ditunjukkan Tabel 3.9. Table 3.10 merupakan saran yang didapatkan dari *para play tester* setelah memainkan *paper prototyping* yang kedua. Dari beberapa saran yang didapatkan dapat diketahui bahwa *rule* dari *game* sudah bisa diterima karena umumnya saran dari *play tester* lebih mengarah ke *design*. Maka paper prototype bisa diimplementasikan ke digital dengan memperhatikan saran yang diberikan oleh para *play tester*.

## *3.4* Implementasi *Game*

Setelah melakukan perancangan *paper prototyping*, akan dilanjutkan ke tahap implementasi *game* dalam bentuk digital menggunakan *game engine* Unity 3D sesuai dengan hasil perancangan dan evaluasi dari peracangan *paper prototyping.* Pada proses ini akan dijelaskan empat tahapan yaitu spesifikasi *system* yang digunakan, implementasi *gameplay* dan implementasi *level*.

### 3.4.1 *Spesifikasi System*

Spesifikasi sistem *game* ini dikembangkan dalam lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak.

**1. Spesifikasi Perangkat Keras**

Spesifikasi perangkat keras yang digunakan dalam proses pengembangan aplikasi akan dijelaskan pada Tabel 3.9.

**Table 3. 11 Spesifikasi Perangkat Keras**

| No | PERSONAL COMPUTER (PC) | |
|---|---|---|
| 1 | *Processor* | AMD A6-3420M APU (4CPUs), ~ 1.5GHz |
| 2 | *Memory (RAM)* | 4096 MB |
| 3 | *Harddisk* | 500 GB |
| 4 | *Graphic Card* | ATI Radeon 7670M 1GB |

**2. Spesifikasi Perangkat Lunak**

Spesifikasi perangkat lunak yang digunakan dalam proses pengembangan aplikasi akan dijelaskan pada Tabel 3.10.

**Table 3.12 Spesifikasi Perangkat Lunak**

| No | ***SOFTWARE*** | |
|---|---|---|
| 1 | *Operating System* | Windows 7 Ultimate 64-bit (6.3, Build 7600) |
| 2 | *DirectX Version* | DirectX 11 |
| 3 | *Programming Language* | C# |
| 4 | *Software Development Kit* | *Game Engine* Unity *Version* 4.6 |

### 3.4.2 Implementasi *Gameplay*

Pada tahap implementasi *gameplay* ini dibagi menjadi 3 tahapan yaitu implementasi prosedur program, implementasi *art* dan UI serta simulasi *gameplay*.

#### 3.4.2.1 Implementasi Prosedur Program

*Game* ini terbentuk dari berbagai macam proses atau *method*. Proses yang akan dicantumkan pada penulisan skripsi ini adalah yang paling utama dan paling penting dalam pembuatan game ini.

**1. Implementasi Prosedur Interaksi Pemain**

Prosedur interaksi pemain akan menjelaskan proses dari apa yang bisa dilakukan pemain pada *gameplay* yaitu implementasi interaksi *user* dengan *character* pada *gameplay collecting item* dan *item match*

- **Implementasi Item Controller Pada Collecting Item**

**Table 3.13 Method HandleInput**

| No | Method HandleInput |
|---|---|
| | **Declaration :** |
| | float _normalizedHorizontalSpeed;<br>CharacterController2D _controller; |
| | **Description :** |
| 1 | IF (Input.GetKey(KeyCode.D)) THEN |
| 2 | _normalizedHorizontalSpeed = 1; |
| 3 | IF (!_isFacingRight) THEN |
| 4 | Flip(); |
| 5 | END IF |
| 6 | ELSE IF (Input.GetKey(KeyCode.A)) THEN |
| 7 | _normalizedHorizontalSpeed = -1; |
| 8 | IF (_isFacingRight) THEN |
| 9 | Flip(); |
| 10 | END IF |
| 11 | ELSE |
| 12 | _normalizedHorizontalSpeed = 0; |
| 13 | END IF |
| 14 | IF (_controller.CanJump && Input.GetKeyDown(KeyCode.Space)) THEN |
| 15 | _controller.Jump(); |
| 16 | END IF |

Pada baris 1, berfungsi untuk mengecek apakah tombol keyboard "D" aktif atau tidak. Baris dua untuk memberikan nilai pada variable _normalizedHorizontalSpeed, nilai ini yang akan menentukan *character* berjalan ke kiri dan kanan, apabila bernilai positif maka akan berjalan ke kanan sebaliknya jika bernilai negatif akan bergerak ke kiri. Baris 3 untuk melakukan pengecekan apakah *character* tidak menghadap ke kanan, jika bernilai *true* maka akan dijalankan *method* Flip(). Baris 6 berfungsi untuk mengecek apakah tombol keyboard "A" aktif atau tidak. Baris 14 berfungsi untuk mengecek apakah tombol keyboard "Space" aktif atau tidak dan mengecek nilai variable CanJump dari *class* CharacterController2D jika bernilai *true* maka akan menjalankan *method* Jump().

**Table 3.14 Method Jump**

| No | **Method Jump** |
|---|---|
| | **Declaration :** |
| | public ControllerParameters2D Parameters; |
| | **Description :** |
| | IF (State.IsGround == true) THEN |
| | AddForce (new Vector2 (0, Parameters.JumpMagnitude)); |
| | END IF |

Pada *method* ini berfungsi mengecek apakah *character* berada di daratan atau tidak. Jika ya maka nilai pada sumbu y sama dengan nilai JumpMagnitude yang ada di *class* ControllerParameters2D.

**Table 3.15 Method Move**

| No | **Method Move** |
|---|---|
| | **Declaration :** |
| | var wasGrounded = State.IsCollidingBelow;<br>Vector2 deltaMovement;<br>Vector3 _activeGlobalPlatformPoint, _activeLocalPlatformPoint;<br>bool HandleCollisions = true;<br>ControllerState2D State; |
| | **Description :** |
| 1 | var wasGrounded = State.IsCollidingBelow; |
| 2 | State.Reset(); |
| 3 | |
| 4 | IF (HandleCollisions) THEN |
| 5 | HandlePlatforms(); |
| 6 | CalculateRayOrigins(); |
| 7 | |
| 8 | IF (Mathf.Abs(deltaMovement.x) > .001f) THEN |
| 9 | MoveHorizontally(ref deltaMovement); |
| 10 | END IF |

**Table 3.16 Method Move (lanjutan)**

| | |
|---|---|
| 11 | |
| 12 | MoveVertically(ref deltaMovement); |
| 13 | END IF |
| 14 | |
| 15 | _transform.Translate(deltaMovement, Space.World); |
| 16 | |
| 17 | IF (Time.deltaTime > 0) THEN |
| 18 | _velocity = deltaMovement / Time.deltaTime; |
| 19 | END IF |
| 20 | |
| 21 | _velocity.x = Mathf.Min(_velocity.x, Parameters.MaxVelocity.x); |
| 22 | _velocity.y = Mathf.Min(_velocity.y, Parameters.MaxVelocity.y); |
| 23 | |
| 24 | IF (StandingOn != null) THEN |
| 25 | _activeGlobalPlatformPoint = transform.position; |
| 26 | _activeLocalPlatformPoint = StandingOn.transform.InverseTransformPoint(transform.position); |
| 27 | |
| 28 | IF (_lastStandingOn != StandingOn) THEN |
| 29 | IF (_lastStandingOn != null) THEN |
| 30 | _lastStandingOn.SendMessage("ControllerExit2D", this, SendMessageOptions.DontRequireReceiver); |
| 31 | END IF |
| 32 | |
| 33 | StandingOn.SendMessage("ControllerEnter2D", this, SendMessageOptions.DontRequireReceiver); |
| 34 | _lastStandingOn = StandingOn; |
| 35 | ELSE IF (StandingOn != null) THEN |
| 36 | StandingOn.SendMessage("ControllerStay2D", this, SendMessageOptions.DontRequireReceiver); |
| 37 | END IF |
| 38 | |
| 39 | ELSE IF (_lastStandingOn != null) THEN |
| 40 | _lastStandingOn.SendMessage("ControllerExit2D", this, SendMessageOptions.DontRequireReceiver);; |
| 41 | _lastStandingOn = null; |
| 42 | END IF |

Pada baris 1-2 akan memberikan nilai kepada variable wasGround sama dengan nilai IsCollidingBelow di *class* ControllerState2D. Kemudian akan menjalankan fungsi Reset() yang akan me-*reset* semua nilai di *class* ControllerState2D. Baris 4-13 akan dijalankan jika HandleCollisions bernilai *true*, jika bernilai *true* maka akan menjalankan *method* HandlePlatform(), CalculateRayOrigins(), MoveVertically() dan *method* MoveHorizontally() apabila nilai mutlak dari deltaMovement.x lebih besar dari 0.001. Baris 15 berfungsi membuat *character* berpindah tempat relative terhadap koordinat dunia *game* itu sendiri. Baris 17 – 22 merupakan kode tambahan apabila *character* berdiri pada *platform* yang bergerak. Baris 24 – 42 untuk memeriksa apakah *character* berdiri pada *platform* atau tidak.

- **Implementasi Item Controller Pada Item Match**

**Table 3.17 Method onMouseDown**

| No | Method OnMouseDown |
|---|---|
| | **Declaration :** |
| | handleToOriginVector, pos |
| | **Description :** |
| 1 | handleToOriginVector = transform.root.position - Camera.main.ScreenToWorldPoint (Input.mousePosition); |
| 2 | pos = transform.root.position; |

Pada baris 1, variabel handleToOriginVector untuk menyimpan selisih nilai posisi awal dari *game object* dikurang posisi ketika *user* melakukan *click input* pada *game object* sesuai dengan *screen* yang dipakai. Sedangkan pada baris 2 pos menyimpan posisi *game object* sesuai dengan *screen* yang dipakai. *Method* ini berfungsi untuk menentukan dan menyimpan posisi *game object item* ketika *user* menekan *item*.

**Table 3.18 Method OnMouseDrag**

| No | Method OnMouseDrag |
|---|---|
| | **Declaration :** |
| | |
| | **Description :** |
| 1 | transform.root.position = Camera.main.ScreenToWorldPoint (Input.mousePosition) + handleToOriginVector; |

Pada method ini baris 1 dijalankan ketika *user* melakukan *drag* pada *gameobject item*, maka akan melakukan inisialisasi nilai pada Input.mousePosition yang menyimpan nilai baru pada inputan posisi *mouse*. Kemudian nilai ini akan di tambah dengan nilai variable handleToOriginVector yang didapatkan dari *method* OnMouseDown. Hasil penjumlahan tersebut akan dimasukkan pada transform.root.position yang berfungsi melakukan perubahan posisi *item* saat user melakukan *drag* ke posisi yang diinginkan.

**Table 3.19 Method OnMouseUp**

| No | **Method OnMouseUp** |
|---|---|
| | **Declaration :** |
| | isFix, |
| | **Description :** |
| | IF (isFix == false) THEN |
| | transform.position = pos; |
| | ELSE |
| | this.gameObject.SetActive(false); |
| | aw.GetComponent<ItemDrop>().Hilang(); |
| | LevelManagerCo.Instance.AddItemMatch(); |
| | END IF |

Pada *method* ini mengecek apakah isFix bernilai *false* apa tidak, jika ya maka nilai transform.position sama dengan nilai variable pos yang didapatkan dari *method* OnMouseDown yang akan membuat *item* kembali ke posisi semula. Tapi apabila bernilai *true* maka akan membuat *gameobject* akan hilang, begitu juga dengan *gameobject* yang memiliki komponen *class* ItemDrop. Kemudian akan menjalankan fungsi AddItemMatch() yang berada di *class* LevelManagerCo untuk menambahkan nilai berapa *item* yang sudah benar.

**Table 3.20 Method OnTriggerStay2D**

| No | **Method OnTriggerStay2D** |
|---|---|
| | **Declaration :** |
| | Int dropid |
| | **Description :** |
| 1 | IF (other.GetComponent<ItemDrag>().id == dropid) THEN |
| 2 | other.GetComponent<ItemDrag>().Aktif(); |
| 3 | END IF |

Pada *method* OnTriggerStay2D dilakukan pengecekan jika id *gameobject* yang memiliki komponen *class* ItemDrag sama dengan nilai dropid maka akan menjalankan fungsi Aktif() pada *class* ItemDrag yang akan membuat nilai isFix bernilai *true*.

**Table 3.21 Method OnTriggerExit2D**

| No | Method OnTriggerStay2D |
|---|---|
| | **Declaration :** |
| | |
| | **Description :** |
| 1 | other.GetComponent<ItemDrag>().Mati(); |

Pada *method* OnTriggerExit2D akan menjalankan fungsi Mati() pada *class* ItemDrag yang akan membuat variable isFix bernilai *false*.

**2. Implementasi Prosedur Penyimpanan Data**

**Table 3.22 Method Save**

| No | Method save |
|---|---|
| | **Declaration :** |
| | int levelIndex,<br>string currentLevel |
| | **Description :** |
| 1 | FOR (int level = 1; level < LockedLevel.levels; level++){ |
| 2 | IF (currentLevel == "Level "+level.ToString()+"_1") THEN |
| 3 | levelIndex = (level+1); |
| 4 | PlayerPrefs.SetInt("level"+levelIndex.ToString(),1); |
| 5 | IF (PlayerPrefs.GetInt("level"+level.ToString()+"stars") < stars) THEN |
| 6 | PlayerPrefs.SetInt("level"+level.ToString()+"stars",stars); |
| 7 | END IF |
| 8 | END IF |
| 9 | END FOR |

Pada *method* ini dari baris 1-4 berfungsi untuk menyimpan data *level* yang sudah dimainkan, *level* yang sudah dimainkan akan di setInt 1. Kemudian baris 5-6. Untuk menyimpan jumlah *stars* yang didapatkan, apabila jumlah *stars* yang didapat sekarang lebih rendah daripada sebelumnya maka jumlah *stars* akan sama dengan jumlah *stars* sebelumnya.

**Table 3.23 Method Save**

| No | Method load |
|---|---|
| | **Declaration :** |
| | int stars, _levelindex |
| | **Description :** |
| 1 | FOR (int j = 1; j < LockedLevel.levels; j++){ |
| 2 | stars = PlayerPrefs.GetInt("level"+j.ToString()+"stars"); |
| 3 | _levelIndex = (j+1); |
| 4 | GameObject.Find(j+"stars"+stars).GetComponent<Image>().enabled = true; |
| 5 | |
| 6 | IF(PlayerPrefs.GetInt("level"+_levelIndex.ToString()) == 1)THEN |
| 7 | GameObject.Find("LockedLevel"+_levelindex).active = false; |
| 8 | END IF |
| 9 | END FOR |

Pada *method* ini pada baris 2 berfungsi untuk mendapatkan nilai *stars* dari *level* yang sudah diselesaikan. Baris 4 untuk memunculkan *stars* yang telah didapatkan. Baris 6-8 untuk membuka *level* yang sudah diselesaikan.

### 3. Implementasi Prosedur How to Win

**Table 3.24 Method HowToWin**

| No | Method howToWin |
|---|---|
| | **Declaration :** |
| | int itemMatch, item<br>float totalTime |
| | **Description :** |
| 1 | IF (itemMatch < item) THEN |
| 2 | totalTime -= Time.deltaTime; |
| 3 | IF (totalTime <= 0) THEN |
| 4 | totalTime = 0; |
| 5 | StartCoroutine(GameOver()); |
| 6 | END IF |
| 7 | timerText.text = "TIME: "+totalTime.ToString("0"); |
| 8 | ELSE |
| 9 | StartCoroutine(Finished()); |
| 10 | END IF |

Pada *method* ini dilakukan pengecekan apabila jumlah itemMatch kurang dari *item* maka akan dijalankan totalTime akan dikurangi dengan jumlah waktu yang berjalan. Apabila nilai totalTime kurang dari 0 maka akan memunculkan *screen game over*. Jika nilai itemMatch sama dengan atau lebih dari nilai item akan memunculkan *screen game win*.

### 3.4.2.2 Implementasi Art dan UI

Implementasi *art* terdiri dari implementasi gambar bangun datar yang digunakan pada lingkungan *game*, *character, tiles,* dan *background*.

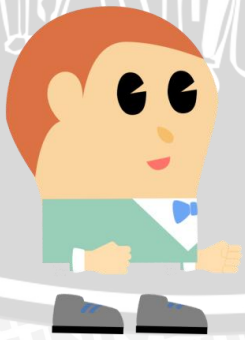
**1. Implementasi *Art* Bangun Datar**

Implemetasi *art* bangun datar ditunjukkan pada gambar dibawah ini

**Gambar 3. 18 Bangun Datar**

**2. Implementasi *Art Character***

Implemetasi a*rt character* ditunjukkan pada gambar dibawah ini



**Gambar 3. 19 *Character***

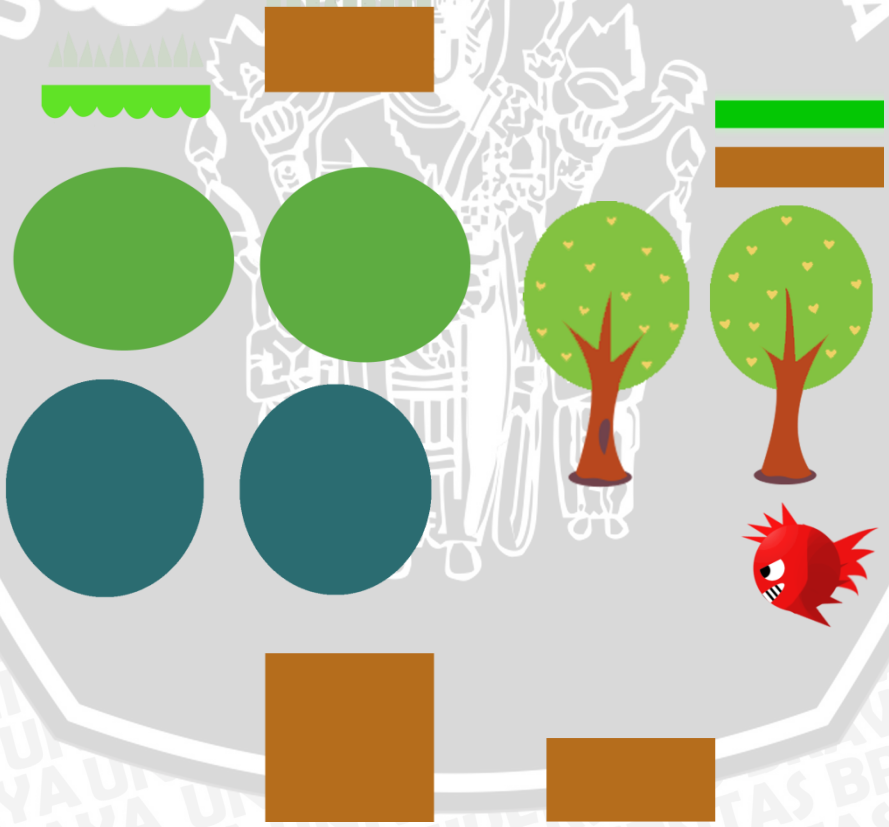
3. **Implementasi *Art Icon***

Implemetasi *art icon* ditunjukkan pada gambar dibawah ini

**Gambar 3. 20 Icon**

4. **Implementasi *Art Texture***

Implemetasi *art texture* ditunjukkan pada gambar dibawah ini



**Gambar 3. 21 Texture**

5. **Implementasi *Game Screen***

Implementasi *game screen* dibagi menjadi 4 yaitu halaman utama, halaman *tutorial* dan halaman pilihan *level*.

- **Implementasi Halaman Utama**

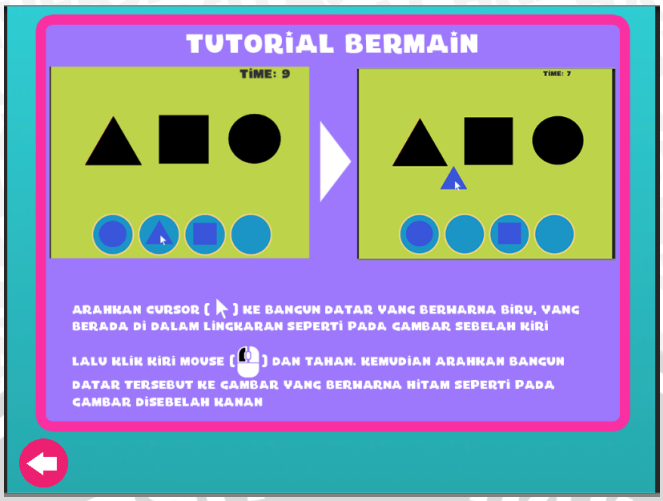
Implementasi antarmuka halaman utama ditunjukan pada Gambar 3.22.


**Gambar 3. 22 Realisasi *Screen* Halaman Utama**

- **Implementasi Halaman *Tutorial***

Implementasi antarmuka halaman *tutorial* ditunjukan pada Gambar 3.23 & 3.24.

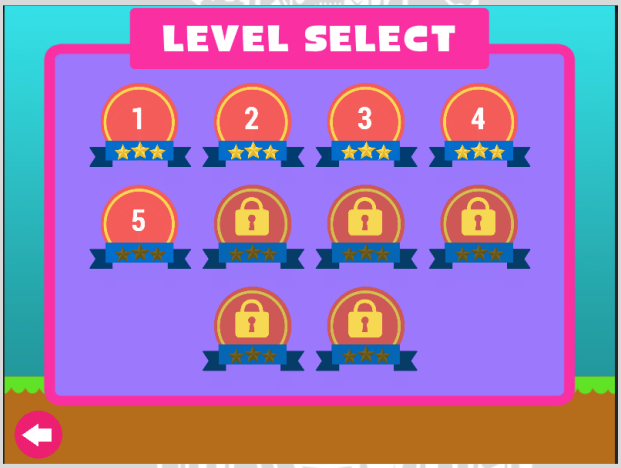

**Gambar 3. 23 Realisasi *Screen Tutorial***




**Gambar 3. 24 Realisasi *Screen Tutorial***

- **Implementasi Halaman Pilihan *Level***

Implementasi antarmuka halaman pilihan *level* ditunjukan pada Gambar 3.25.




**Gambar 3. 25 Realisasi *Screen* Pilih *Level***

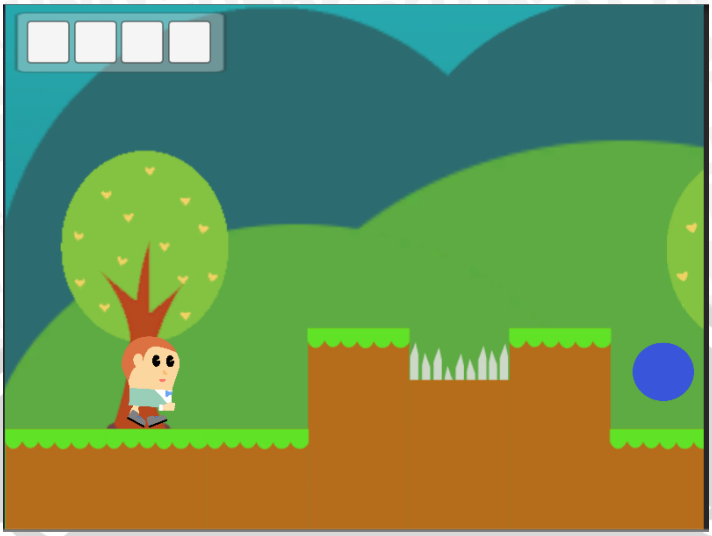
### 3.4.2.3 Simulasi *Gameplay*

Simulasi *gameplay* terdiri dari *gameplay collecting item* dan *item match.*

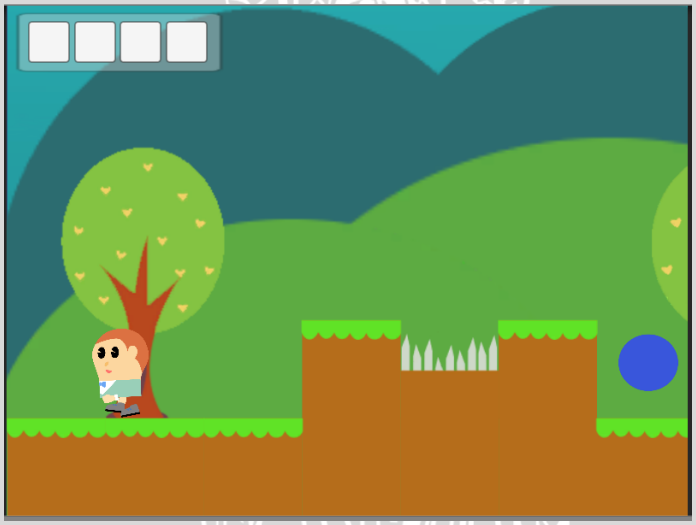
**1. *Gameplay Collecting Item***

Pada tahap ini akan di jelaskan simulasi *gameplay collecting item* yang terdiri dari cara untuk menggerakan *character*, kondisi *character* ketika menyentuh *spike traps,* dan kondisi jika *player* sudah mendapatkan semua *item* dan berada di titik tujuan.

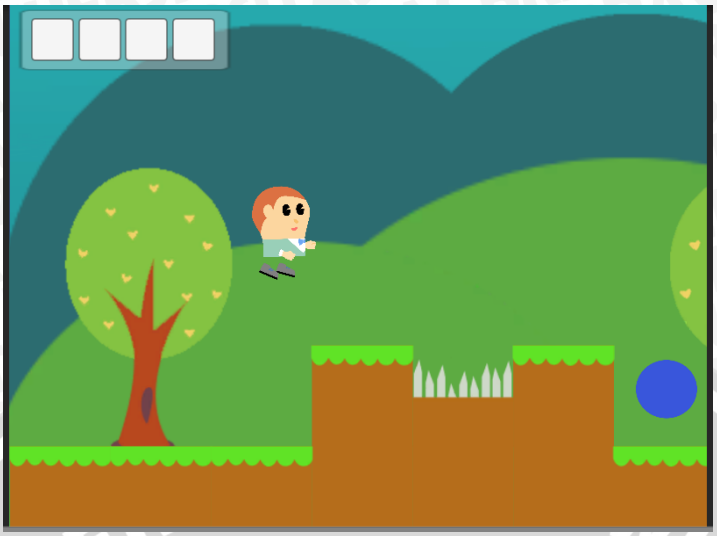
- Menggerakan *Character*



**Gambar 3. 26 Implementasi *Character* Berjalan Ke Kanan**



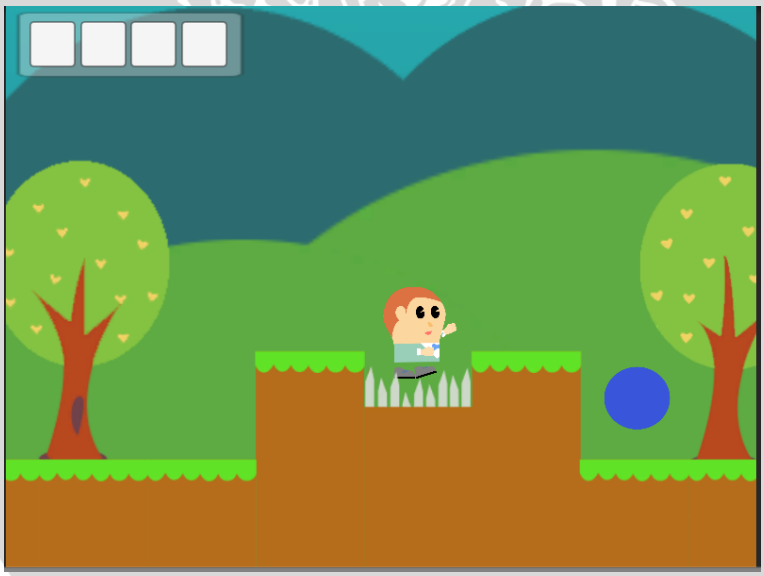
**Gambar 3. 27 Implementasi *Character* Berjalan Ke Kiri**



**Gambar 3. 28 Implementasi *Character* Melompat**

Pada gambar diatas mensimulasikan bagaimana *character* berjalan ke kanan, ke kiri, dan melompat seperti ditunjukkan pada Gambar 3.26, 3.27, & 3.28.

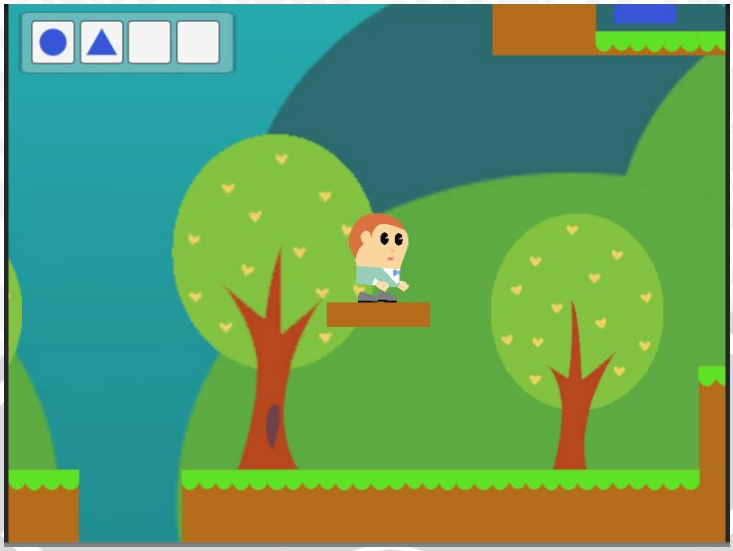
- **Kondisi *Character* Menyentuh Spike Traps**



**Gambar 3. 29 Implementasi *Character* Menyentuh *Spike Traps***

Pada gambar diatas mensimulasikan kondisi *character* ketika menyentuh *spike traps*. Maka *character* akan mati dan kembali ke tempat memulai permainan.

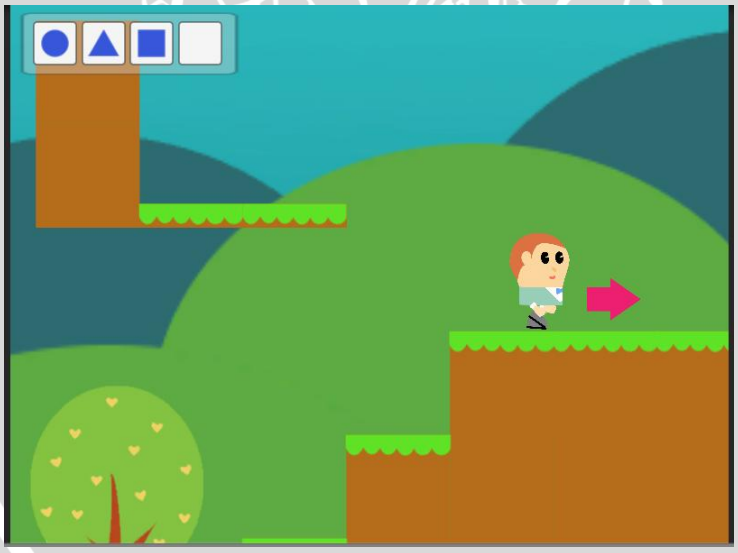
- **Kondisi *Character* Menyentuh Item**



**Gambar 3. 30 Implementasi *Character* Menyentuh *Item***

Pada gambar diatas mensimulasikan kondisi *character* ketika menyentuh *item*. Maka *item* akan hilang dan muncul di *box* sebelah kiri atas layar.

- **Kondisi *Character* menyentuh tanda panah ke kanan**



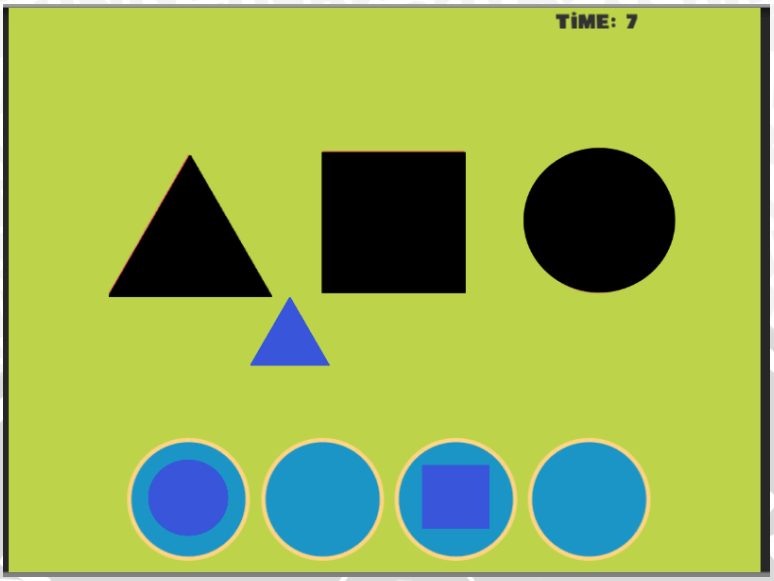
**Gambar 3. 31 Implementasi *Character* Menyentuh Tanda Panah ke Kanan**

Pada gambar diatas mensimulasikan kondisi *character* ketika menyentuh tanda panah ke kanan. Maka *player* akan berindah ke *game screen item match.*

**2. *Gameplay Item Match***

Pada tahap ini akan di jelaskan simulasi *gameplay item match* yang terdiri dari cara untuk menggerakan *item*, kondisi *item* ketika menyentuh bangun datar berwarna hitam, dan kondisi jika *player* menang dan kalah.

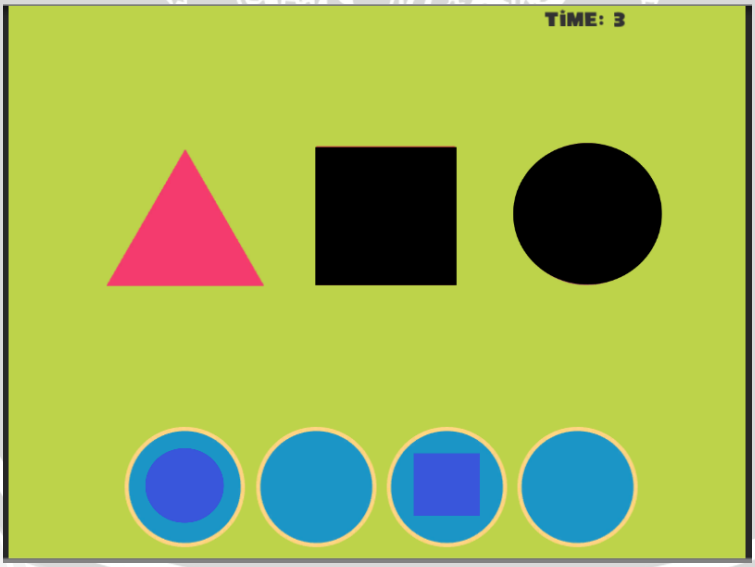
- **Menggerakan Item**




**Gambar 3. 32 Implementasi Menggerakan *Item***

Pada gambar diatas mensimulasikan bagaimana menggerakkan *item* yang ditunjukan pada Gambar 3.32.

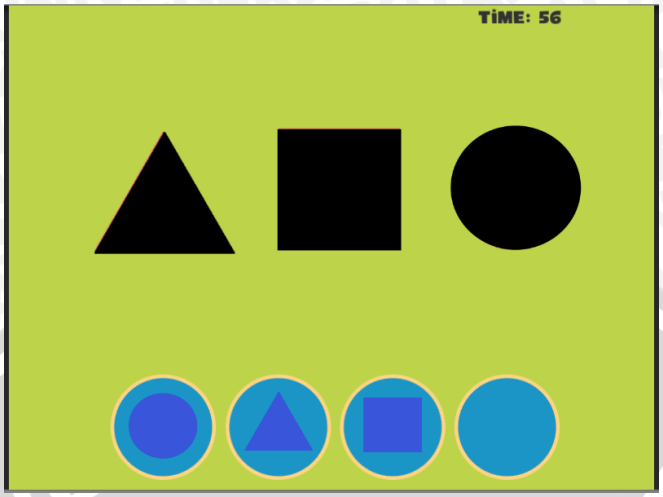
- **Kondisi Jawaban Benar**




**Gambar 3. 33 Implementasi Jika Jawaban Benar**

Pada gambar diatas mensimulasikan bagaimana jika *item* yang dipasangkan benar. Bentuk *item* dan gambar cocok maka gambar yang berwarna hitam akan hilang dan akan berwarna seperti pada Gambar 3.33.

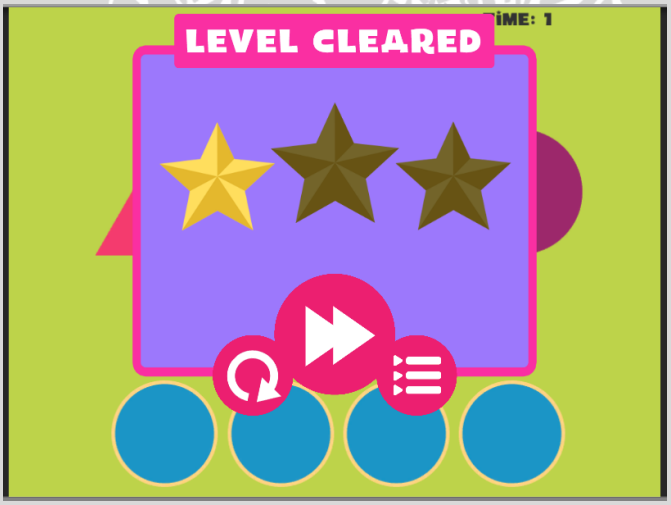
- **Kondisi Jawaban Salah**




**Gambar 3. 34 Implementasi Jika Jawaban Salah**

Pada gambar diatas mensimulasikan bagaimana jika *item* yang dipasangkan salah. Jika salah maka posisi *item* akan kembali seperti semula seperti pada Gambar 3.34.

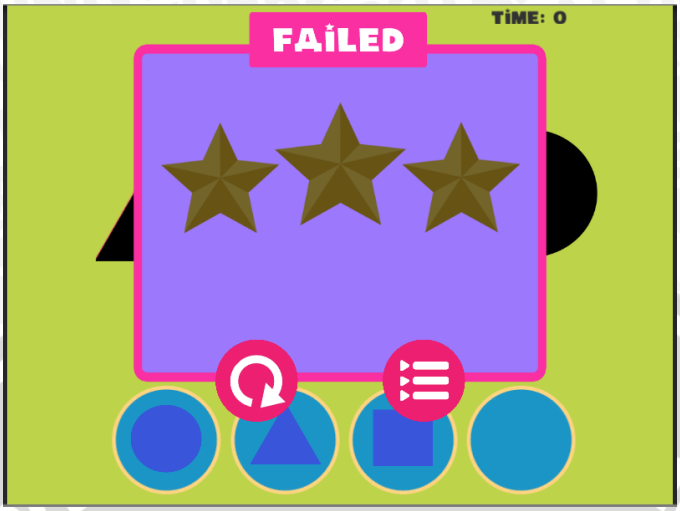
- **Kondisi Menang**




**Gambar 3. 35 Implementasi Kondisi Menang**

Pada gambar diatas mensimulasikan bagaimana jika *player* berhasil menyelesaikan permainan. Permainan diaggap selesai jika semua *item* berhasil dipasangkan.

- **Kondisi Kalah**




**Gambar 3. 36 Implementasi Kondisi Kalah**

Pada gambar diatas mensimulasikan bagaimana jika *player* tidak dapat menyelesaikan permainan sebelum waktu yang ditentukan maka *player* akan dianggap kalah.

### 3.4.3 Implementasi *Level*

Pada tahap ini akan menampilkan hasil implementasi sepuluh *level* pada *game* ini. Setiap level memiliki 2 *gameplay* yaitu *gameplay collecting item* dan *item match.*

**Table 3.25 Realisasi *Level***

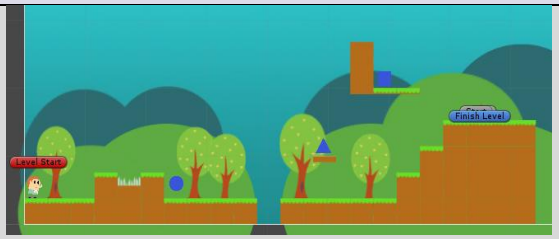
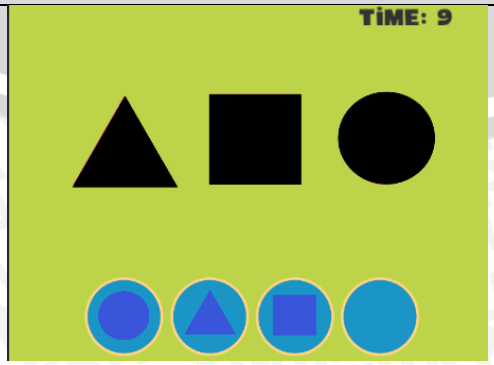
| Realisasi Level | | | Keterangan |
|---|---|---|---|
| Level 1 | Collecting Item |  | User akan mencari item dengan melewati rintangan dan *hazard*. |
| | Item Match |  | User akan mencocokan *item* dengan gambar yang berwarna hitam. Waktu 60 detik. |

**Table 3.26 Realisasi *Level* (lanjutan)**

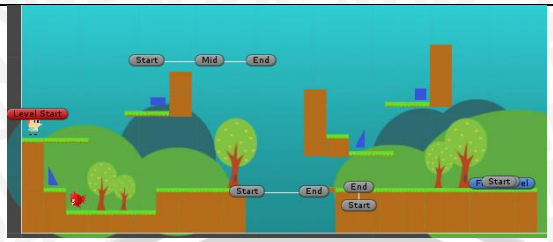
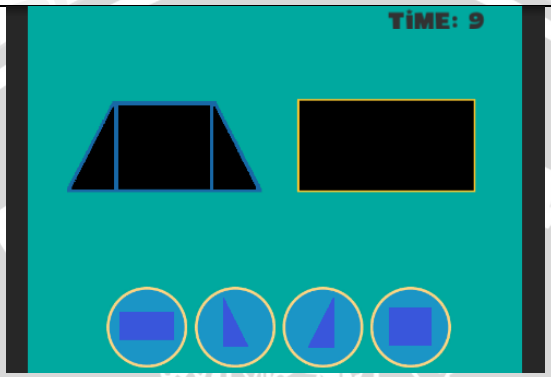
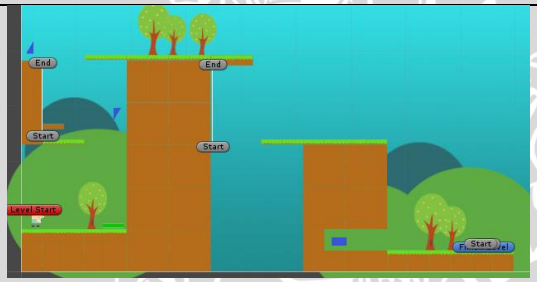
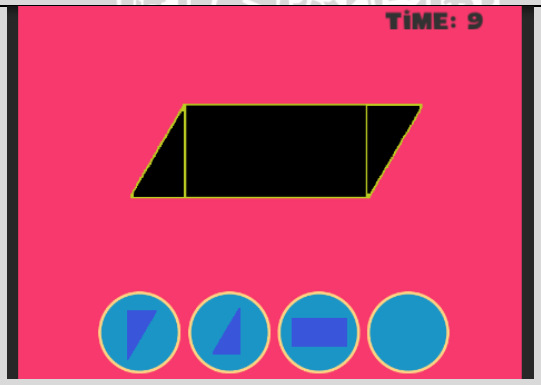
| | | | |
|---|---|---|---|
| **Level 2** | **Collecting Item** |  | User akan mencari *item* dengan melewati rintangan dan *hazard*. |
| | **Item Match** |   | User akan mencocokan *item* dengan gambar yang berwarna hitam. Waktu 60 detik. |
| **Level 3** | **Collecting Item** |  | User akan mencari *item* dengan melewati rintangan dan *hazard*. |
| | **Item Match** |   | User akan mencocokan *item* dengan gambar yang berwarna hitam. Waktu 60 detik. |

**Table 3.27 Realisasi *Level* (lanjutan)**

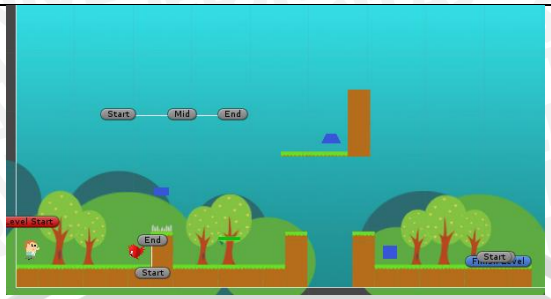
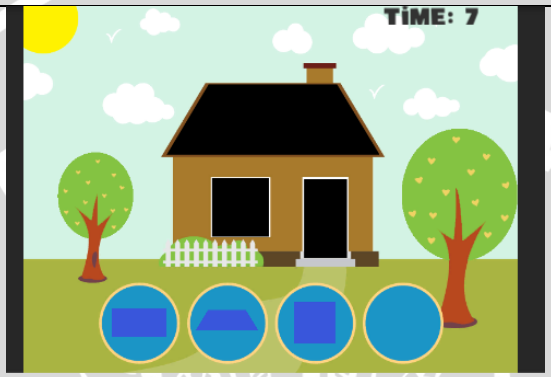
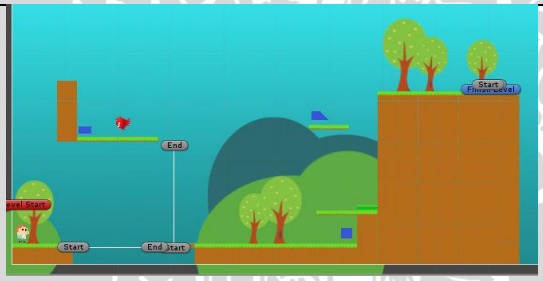
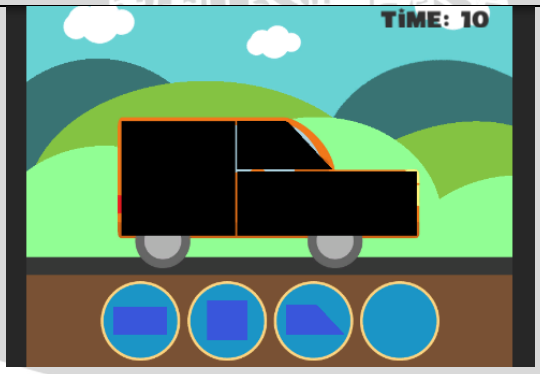
| | | | |
|---|---|---|---|
| **Level 4** | **Collecting Item** |  | User akan mencari *item* dengan melewati rintangan dan *hazard*. |
| | **Item Match** |  | User akan mencocokan *item* dengan gambar yang berwarna hitam. Waktu 60 detik. |
| **Level 5** | **Collecting Item** |  | User akan mencari *item* dengan melewati rintangan dan *hazard*. |
| | **Item Match** |  | User akan mencocokan *item* dengan gambar yang berwarna hitam. Waktu 60 detik. |

**Table 3.28 Realisasi *Level* (lanjutan)**

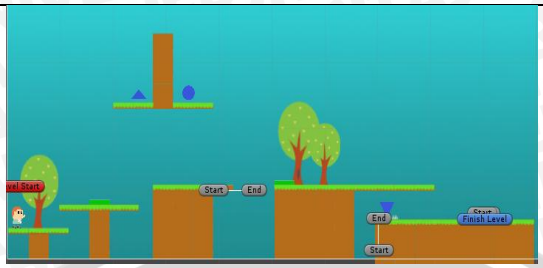
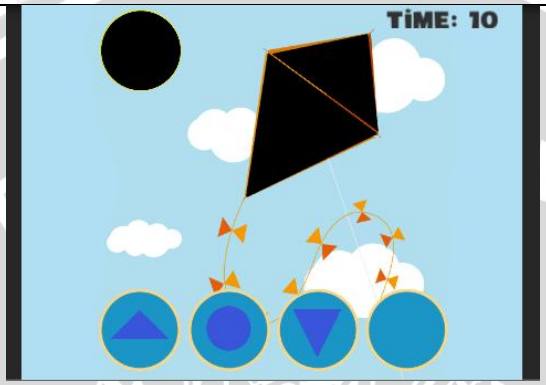
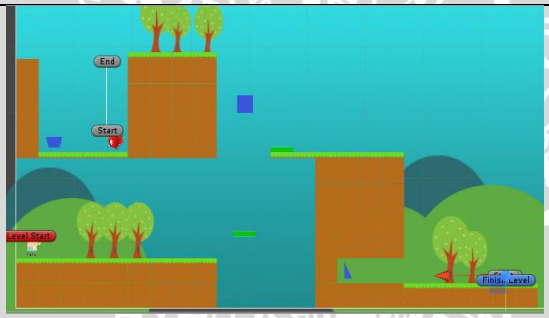
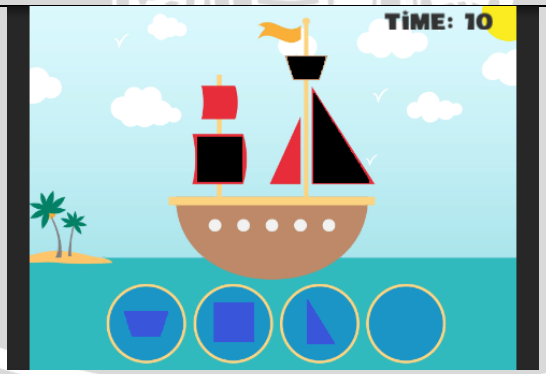
| | | | |
|---|---|---|---|
| **Level 6** | **Collecting Item** |  | User akan mencari *item* dengan melewati rintangan dan *hazard*. |
| | **Item Match** |  | User akan mencocokan *item* dengan gambar yang berwarna hitam. Waktu 60 detik. |
| **Level 7** | **Collecting Item** |  | User akan mencari *item* dengan melewati rintangan dan *hazard*. |
| | **Item Match** |  | User akan mencocokan *item* dengan gambar yang berwarna hitam. Waktu 60 detik. |

**Table 3.29 Realisasi *Level* (lanjutan)**

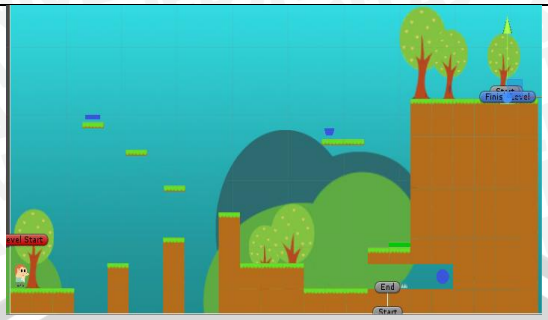
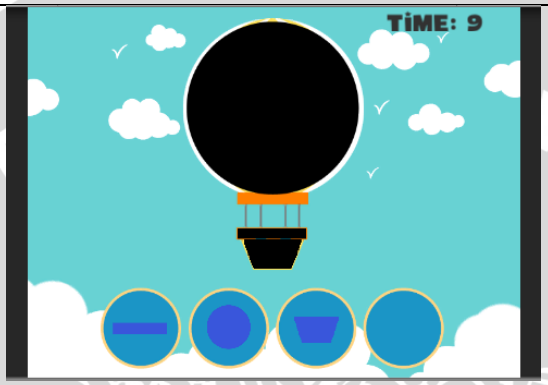
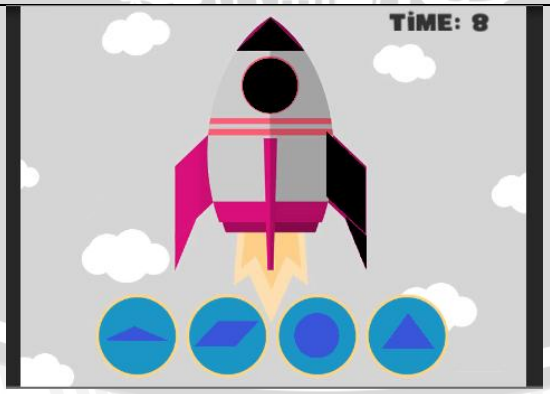
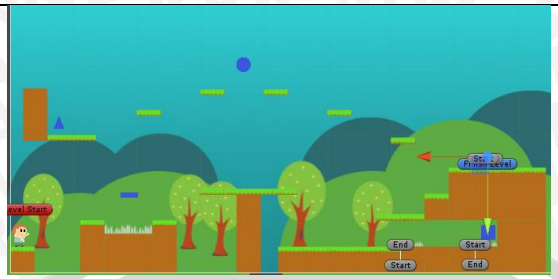
| | | | |
|---|---|---|---|
| **Level 8** | **Collecting Item** |  | User akan mencari *item* dengan melewati rintangan dan *hazard*. |
| | **Item Match** |  | User akan mencocokan *item* dengan gambar yang berwarna hitam. Waktu 60 detik. |
| **Level 9** | **Collecting Item** |  | User akan mencari *item* dengan melewati rintangan dan *hazard*. |
| | **Item Match** |  | User akan mencocokan *item* dengan gambar yang berwarna hitam. Waktu 60 detik. |

**Table 3.30 Realisasi *Level* (lanjutan)**

| | | | |
|---|---|---|---|
| **Level 10** | **Collecting Item** |  | User akan mencari *item* dengan melewati rintangan dan *hazard*. |
| | **Item Match** |  | User akan mencocokan *item* dengan gambar yang berwarna hitam. Waktu 60 detik. |

## 3.5 Pengujian

Setelah melakukan implementasi *game*, akan dilanjutkan ke tahap pengujian untuk menguji *game* edukasi yang telah dibuat. Pada pengujian yang akan dilakukan menggunakan dua metode pengujian yaitu *white box* dan *black box.* Dalam melakukan pengujian *white box* digunakan teknik *basis path testing* sedangkan untuk pengujian *black box* digunakan 2 metode yaitu *test flow diagram* dan *focus testing*.

Pada teknik *basis path testing* proses pengujian dilakukan dengan memodelkan algoritma pada sebuah *flow graph*, menentukan *cylometic complexity* dan melakukan uji kasus untuk setiap *path* yang ada. Pada pengujian *test flow diagram* proses pengujian ini memberikan pendekatan formal untuk desain tes yang mempromosikan modularitas dan kelengkapan dari *game* yang akan diuji (proses dalam satu *game*), yang bertujuan agar mudah untuk ditinjau, dianalisis dan mendapatkan *feedback* (umpan balik) pada desain yang akan diuji bagi penguji. Sedangkan pada *focus testing* dilakukan untuk memastikan bahwa pembuatan *game* telah sesuai dengan tujuan yang ingin dicapai yaitu *game* yang menyenangkan dan interaktif yang dapat digunakan sebagai media pembelajaran baru untuk memperkenalkan bangun datar 2D.

# BAB 4 HASIL

Pada tahap ini akan menyajikan hasil dari pengujian yang telah dilakukan pada *game* edukasi "Shape Adventure". Hasil pengujian dibagi menjadi dua yaitu *white box* dan *black box*.

## 4.1 *White Box*

Pada tahap *white box testing* digunakan pengujian dengan teknik *basis path testing.* Pada penulisan laporan skripsi ini hanya dicantukam hasil *basis path testing* untuk prosedur dari beberapa metode saja atau hanya mencantumkan metode yang penting atau intinya saja (tidak untuk keseluruhan metode).

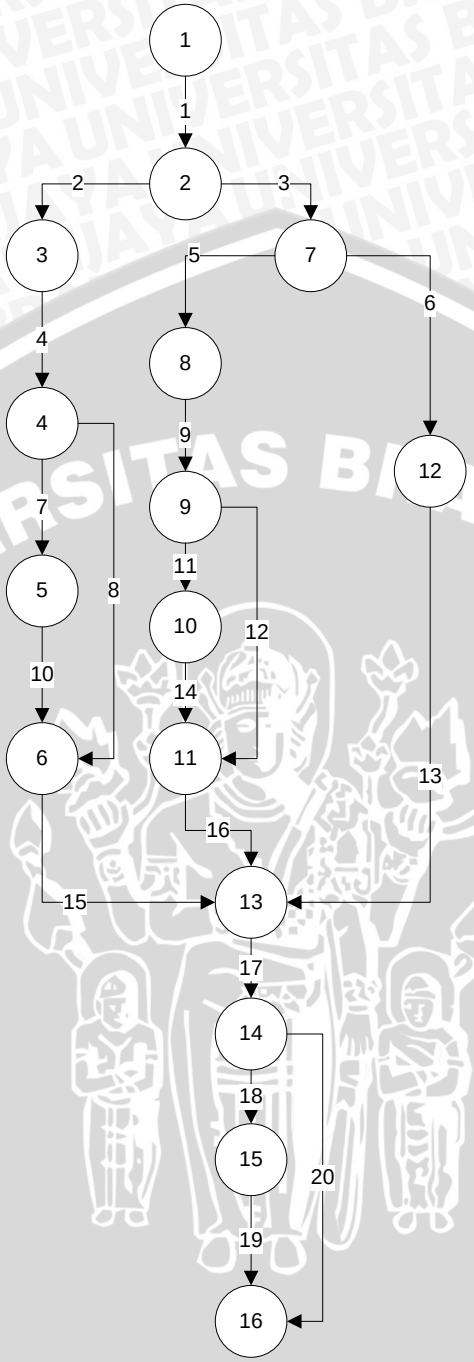
**1. Pengujian Basis Path Menggerakkan *Character***

**Pseudocode**
**Menggerakkan *character* Pada *Collecting Item***

```
Deklarasi awal :
float _normalizedHorizontalSpeed,
CharacterController2D _controller
Deskripsi :
START
  IF (Input.GetKey(KeyCode.D)) THEN
    _normalizedHorizontalSpeed = 1;
    IF (!_isFacingRight) THEN
       Flip()
    END IF
  ELSE IF (Input.GetKey(Ke                )) THEN
    _normalizedHorizontalSpeed = -1;
          IF (_isFacingRight) THEN
       Flip()
    END IF
  ELSE
    _normalizedHorizontalSpeed = 0;
  END IF

  IF (Input.GetKeyDown(KeyCode.Space)) THEN
    _controller.Jump();
  END IF
END
```

**Gambar 4. 1 Pengujian *Basis Path Controller Character***




**Gambar 4. 2 *Flow Graph* Pengujian *Basis Path Controller Character***

Pada pemodelan algoritma ke dalam *flow graph* yang telah dilakukan untuk mengontrol *character* pada *collecting item* menghasilkan jumlah kompleksitas siklomatis (*cyclomatic complexity)* melalui persamaan V(G) = E-N+2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node),* N merupakan jumlah simpul (*node*) dan P merupakan jumlah *node* yang memiliki *exit point*

V(G) = E – N + P

= 20 – 16 + 2

= 6

Berdasakan nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka ditentukan 6 buah basis set dari jalur independent, yaitu :

Jalur 1 : 1-2-3-4-5-6-13-14-16

Jalur 2 : 1-2-3-4-6-13-14-16

Jalur 3 : 1-2-7-8-9-10-11-13-14-16

Jalur 4 : 1-2-7-8-9-11-13-14-16

Jalur 5 : 1-2-7-12-13-14-15-16

Jalur 6 : 1-2-3-4-5-6-13-14-15-16

**Table 4. 1 Pengujian *Basis Path* Mengontrol *character***

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Menekan tombol D pada keyboard ketika *character* menghadap kanan | *Character* berjalan ke kanan | *Character* berjalan ke kanan |
| 2 | Menekan tombol D pada keyboard ketika *character* menghadap kiri | *Character* akan berbalik menghadap kanan kemudian berjalan ke kanan | *Character* akan berbalik menghadap kanan kemudian berjalan ke kanan |
| 3 | Menekan tombol A pada keyboard ketika *character* menghadap kiri | *Character* berjalan ke kiri | *Character* berjalan ke kiri |
| 4 | Menekan tombol A pada keyboard ketika *character* menghadap kanan | *Character* akan berbalik menghadap kiri kemudian berjalan ke kiri | *Character* akan berbalik menghadap kiri kemudian berjalan ke kiri |
| 5 | Menekan tombol Space pada *keyboard* | *Character* akan melompat ke atas | *Character* akan melompat atas |
| 6 | Menekan tombol D pada *keyboard* ketika *character* menghadap kanan sambil menekan tombol Space pada *keyboard* | *Character* akan melompat ke atas dan bergerak ke kanan | *Character* akan melompat ke atas dan bergerak ke kanan |

**2. Pengujian basis path Pencocokan Item Pada Item Matching**

**Pseudocode**
**Pencocokan *item* Pada *Item Matching***

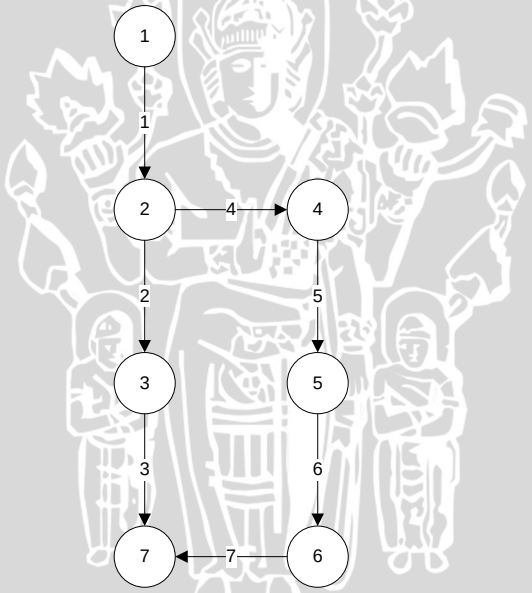
**Deklarasi awal :**
isFix is boolean

**Deskripsi :**

```
START
  IF (isFix == false) THEN
    transform.position = pos;
  ELSE
    this.gameObject.SetActive(false);
    aw.GetComponent<ItemDrop>().Hilang();
    LevelManagerCo.Instance.AddItemMatch();
  END IF
END
```

1 2 3 4 5 6 7

**Gambar 4. 3 Pengujian *Basis Path* Pencocokan Item**




**Gambar 4. 4 *Flow Graph* Pengujian *Basis Path* Pencocokan *Item***

Pada pemodelan algoritma ke dalam *flow graph* yang telah dilakukan untuk pencocokan *item* pada *item matching* menghasilkan jumlah kompleksitas siklomatis (*cyclomatic complexity)* melalui persamaan V(G) = E-N+2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node)* dan N merupakan jumlah simpul (*node*)

V(G) = E – N + 2

= 7 – 7 + 2

= 2

Berdasarkan nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka ditentukan 2 buah basis set dari jalur independent, yaitu :

Jalur 1 : 1-2-3-7

Jalur 2 : 1-2-4-5-6-7

**Table 4. 2 Pengujian *Basis Path* Pencocokan *Item***

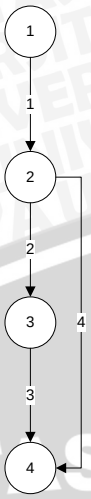
| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Pengecekan *item* jika fix bernilai *false* | *item* kembali ke posisi semula | *item* kembali ke posisi semula |
| 2 | Pengecekan *item* jika fix bernilai *true* | *item* hilang dan gambar hitam menjadi berwarna | *item* hilang dan gambar hitam menjadi berwarna |

### 3. Pengujian basis path Menggeser Item Pada Item Matching

**Pseudocode**
**Menggeser *item* Pada *Item Matching***

**Deklarasi awal :**
isDragging is boolean,
vector3 handleToOriginVector
**Deskripsi :**
START
IF (isDragging == true) THEN
transform.root.position = Camera.main.ScreenToWorldPoint (Input.mousePosition) + handleToOriginVector;
END IF
END

**Gambar 4. 5 Pengujian *Basis Path* Menggeser *Item***




**Gambar 4. 6 *Flow Graph* Pengujian *Basis Path* Menggeser *Item***

Pada pemodelan algoritma ke dalam *flow graph* yang telah dilakukan untuk menggeser *item* pada *item matching* menghasilkan jumlah kompleksitas siklomatis (*cyclomatic complexity)* melalui persamaan V(G) = E-N+2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node)* dan N merupakan jumlah simpul (*node*)

V(G) = E – N + 2

= 4 – 4 + 2

= 2

Berdasarkan nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka ditentukan 2 buah basis set dari jalur independent, yaitu :

Jalur 1 : 1-2-3-4

Jalur 2 : 1-2-4

**Table 4. 3 Pengujian basis path Menggeser *Item***

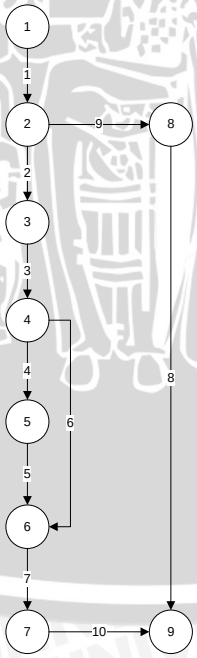
| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Pengecekan *item* jika di drag atau ada inisialisasi nilai pada transform.root.position saat mengeklik kiri mouse | *item* bergerak sesuai posisi inputan player | *item* bergerak sesuai posisi inputan player |
| 2 | Pengecekan *item* jika tidak di drag atau tidak ada inisialisasi nilai pada transform.root.position | Tidak terjadi perubahan posisi *item* | Tidak terjadi perubahan posisi *item* |

## 4. Pengujian basis path Prosedur Menang

**Pseudocode**
**Prosedur Menang**

```
Deklarasi awal :
int itemMatch, item;
float totalTime;
Deskripsi :
START
  IF (itemMatch < item) THEN
        totalTime -= Time.deltaTime;
        IF (totalTime <= 0) THEN
          totalTime = 0;
          StartCoroutine(GameOver());
        END IF
        timerText.text = "TIME: "+totalTime.ToString("0");
  ELSE
        StartCoroutine(Finished());
  END IF
END
```

**Gambar 4. 7 Pengujian *Basis Path* Prosedur Menang**




**Gambar 4. 8 *Flow Graph* Pengujian *Basis Path* Prosedur Menang**

Pada pemodelan algoritma ke dalam *flow graph* yang telah dilakukan untuk prosedur menang menghasilkan jumlah kompleksitas siklomatis (*cyclomatic complexity)* melalui persamaan V(G) = E-N+2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi atau *edge* (garis penghubung antar *node)* dan N merupakan jumlah simpul (*node*)

V(G) = E – N + 2

= 10 – 9 + 2

= 3

Berdasarkan nilai *cyclomatic complexity* yang telah didapatkan dari perhitungan maka ditentukan 3 buah basis set dari jalur independent, yaitu :

Jalur 1 : 1-2-3-4-5-6-7-9

Jalur 2 : 1-2-3-4-6-7-9

Jalur 3 : 1-2-8-9

**Table 4. 4 Pengujian *Basis Path* Pencocokan *Item* Prosedur Menang**

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Pengecekan jumlah *item* yang benar jika kurang dari soal dan waktu lebih dari 0 | *Timer* akan berjalan dan akan berkurang terus sebanyak 1 detik sampai 0 | Waktu akan berkurang terus sebanyak 1 detik sampai 0 |
| 2 | Pengecekan jumlah *item* yang benar kurang dari jumlah soal tetapi waktu kurang dari 0 | Akan menampilkan *screen lose* (karena kehabisan waktu) | Akan menampilkan *screen lose* (karena kehabisan waktu) |
| 3 | Pengecekan jika jumlah *item* yang benar sama dengan jumlah soal | Akan menampilkan *screen Win* | Akan menampilkan *screen Win* |

## 4.2 *Black Box*

Pada tahap *black box testing* ini menggunakan 2 teknik pengujian yaitu pengujian *test flow diagram* dan *focus testing*.

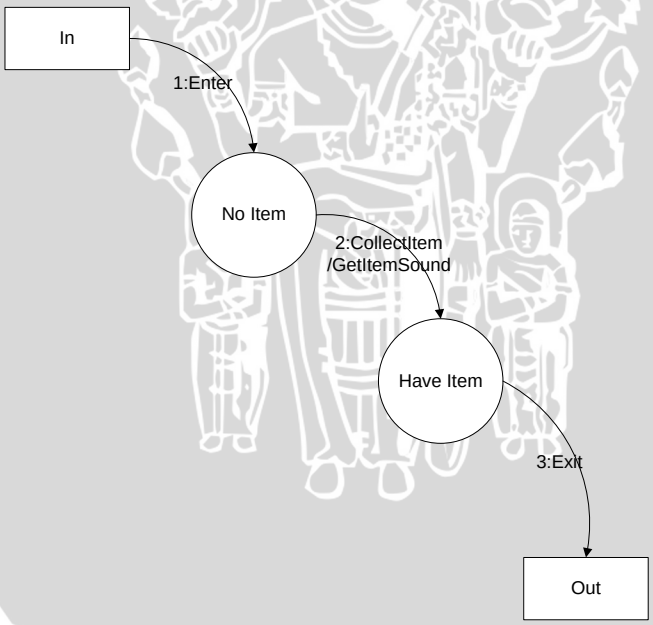
### 4.2.1 Pengujian *Test Flow Diagram*

Pengujian *test flow diagram* (pengujian uji aliran menggunakan/melalui diagram) merupakan model pengujian grafis yang mewakili perilaku pemain dari persepektif pemain/pengguna. Pengujian tersebut memberikan pendekatan formal untuk desain tes yang mempromosikan modularitas dan kelengkapan dari *game* yang akan diuji (proses dalam satu *game*), yang bertujuan agar mudah untuk ditinjau, dianalisis dan mendapatkan *feedback* (umpan balik) pada desain yang akan diuji bagi penguji.

Pada tahap pengujian *test flow diagram* hanya menguji perilaku *game* oleh dari persepektif pemain pada *gameplay collecting item* dan *item match*.

**1. Pengujian *Test Flow Diagram Collecting Item* Jika Mendapatkan *Item***

Berikut pengujian *flow diagram* pada *gameplay collecting item* jika mendapatkan *item*.




**Gambar 4. 9 Pengujian *Test Flow Diagram* Jika Mendapatkan *Item***

Pada tahap ini penghitungan jalur diagram menggunakan strategi penghitungan jalur minimum *path generation*. Kelebihan menggunakan teknik jalur *minimum path generation* ini adalah penguji memiliki jumlah tes yang sedikit dan pengetahuan secara menyeluruh tentang bagian diagram setidaknya satu kali dengan penghitungan jalur sebagai berikut :

- 1-2-3

Berdasarkan perhitungan jalur diatas maka didapatkan satu jalur yang akan dibuat *test case* dengan mengganti kata *primitive* tersebut dengan kamus data sebagai berikut :

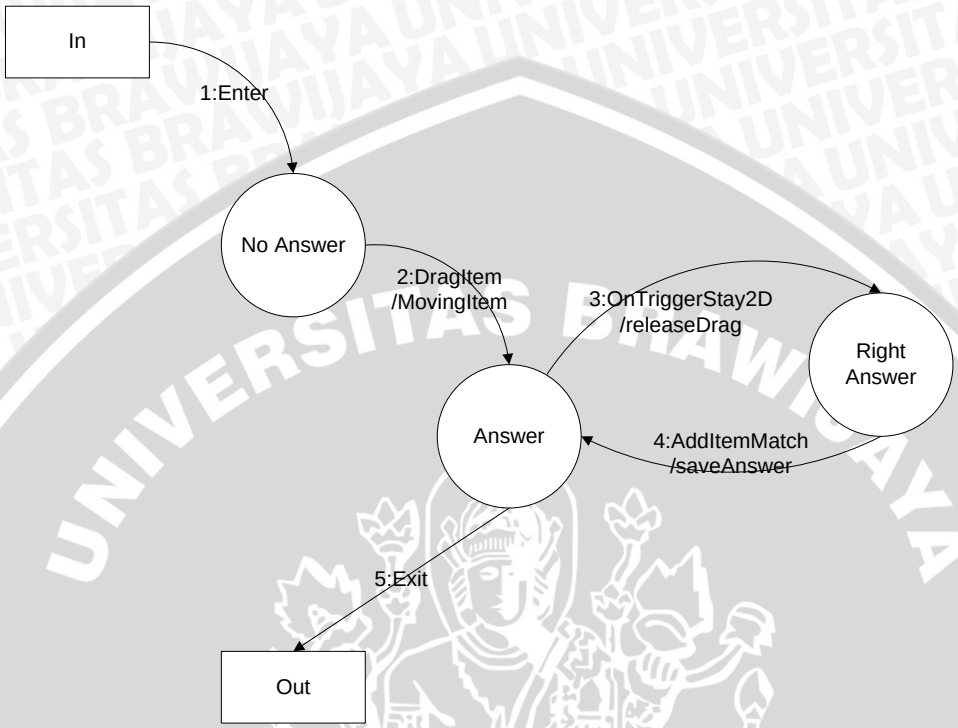
1. Terminator "IN" , Event : *Enter*
   - Menjalankan *game* pada PC
   - Pilih "*play*" kemudian pilih level untuk memulai permainan
2. Event "*CollectItem*", Action : *GetItemSound*
   - Memeriksa apakah *character* menyentuh *item*
   - Memeriksa apakah terdengar suara ketika *item* dan *character* bersentuhan
   - Memeriksa apakah *item* hilang setelah bersentuhan dengan *character*
3. State "*Have Item*"
   - Memeriksa apakah *item* muncul di kotak sebelah pojok kiri atas di dalam screen
4. Event "*Exit*"
   - Mengakhiri proses setelah mendapatkan *item*
5. Terminator "*Out*"
   - Keluar dari kondisi mendapatkan *item*

**Table 4. 5 Pengujian *Test Flow Diagram Collecting Item* Jika Mendapatkan *Item***

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Pengujian prosedur pada *gameplay collecting item* jika mendapat *item* | Menghasilkan hasil yang sesuai pada kebutuhan dan fungsionalitas tanpa ada *bug* pada setiap prosedur jika mendapat *item* pada *gameplay collecting item* | Menghasilkan hasil yang sesuai pada kebutuhan dan fungsionalitas tanpa ada *bug* pada setiap prosedur jika mendapat *item* pada *gameplay collecting item* |

**2. Pengujian *Test Flow Diagram Item Match* Jika Menjawab Benar**

Berikut pengujian *flow diagram* pada *gameplay item match* jika menjawab benar.




**Gambar 4. 10 Pengujian *Test Flow Diagram Item Match* Jika Menjawab Benar**

Pada tahap ini penghitungan jalur diagram menggunakan strategi penghitungan jalur minimum *path generation*. Kelebihan menggunakan teknik jalur *minimum path generation* ini adalah penguji memiliki jumlah tes yang sedikit dan pengetahuan secara menyeluruh tentang bagian diagram setidaknya satu kali dengan penghitungan jalur sebagai berikut :

- 1-2-3-4-5

Berdasarkan perhitungan jalur diatas maka didapatkan 1 jalur yang akan dibuat test case dengan mengganti kata *primitive* tersebut dengan kamus data sebagai berikut :

1. Terminator “IN” , Event : *Enter*
   - Menyelesaikan *gameplay collecting item*
2. Event “*DragItem*”, Action : *MovingItem*
   - Memeriksa posisi *item* jika di drag
   - Memeriksa apakah *item* dapat berpindah ketika digerakkan oleh pemain
3. State “*Answer*”
   - Memeriksa apakah posisi *item* berada pada gambar yang berwarna hitam
4. Event “*OnTriggerStay2D*”, Action : *ReleaseDrag*
   - Memeriksa apakah *item* berada di dalam gambar berwarna hitam

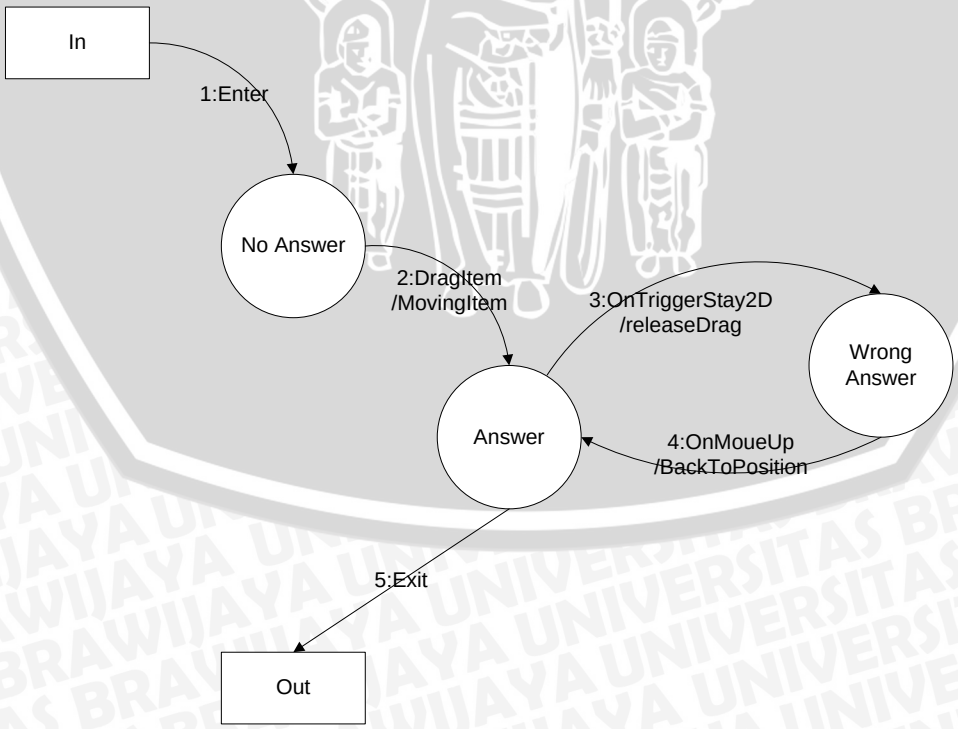
5. State "*Right Answer*"
   - Memeriksa apakah bentuk *item* sama dengan bentuk gambar berwarna hitam
6. Event "*AddItemMatch*", Action : *SaveAnswer*
   - Memeriksa apakah jumlah jawaban yang benar sudah bertambah
7. Terminator "*Out*"
   - Mengakhiri proses jika menjawab benar

**Table 4. 6 Pengujian *Test Flow Diagram Item Match* Jika Menjawab Benar**

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Pengujian prosedur pada *gameplay item match* jika menjawab benar | Menghasilkan hasil yang sesuai pada kebutuhan dan fungsionalitas tanpa ada *bug* pada setiap prosedur jika jika menjawab benar pada *gameplay item match* | Menghasilkan hasil yang sesuai pada kebutuhan dan fungsionalitas tanpa ada *bug* pada setiap prosedur jika jika menjawab benar pada *gameplay item match* |

### 3. Pengujian *Test Flow Diagram Item Match* Jika Menjawab Salah

Berikut pengujian flow diagram pengujian pada *gameplay item match* jika menjawab salah.




**Gambar 4. 11 Pengujian *Test Flow Diagram Item Match* Jika Menjawab Salah**

Pada tahap ini penghitungan jalur diagram menggunakan strategi penghitungan jalur minimum *path generation*. Kelebihan menggunakan teknik jalur *minimum path generation* ini adalah penguji memiliki jumlah tes yang sedikit dan pengetahuan secara menyeluruh tentang bagian diagram setidaknya satu kali dengan penghitungan jalur sebagai berikut :

- 1-2-3-4-5

Berdasarkan perhitungan jalur diatas maka didapatkan 1 jalur yang akan dibuat test case dengan mengganti kata *primitive* tersebut dengan kamus data sebagai berikut :

1. Terminator "IN" , Event : *Enter*
   - Menyelesaikan *gameplay collecting item*
2. Event "*DragItem*", Action : *MovingItem*
   - Memeriksa posisi *item* jika di drag
   - Memeriksa apakah *item* dapat berpindah ketika digerakkan oleh pemain
3. State "*Answer*"
   - Memeriksa apakah posisi *item* berada pada gambar yang berwarna hitam
4. Event "*OnTriggerStay2D*", Action : *ReleaseDrag*
   - Memeriksa apakah *item* berada di dalam gambar berwarna hitam
5. State "*Wrong Answer*"
   - Memeriksa apakah bentuk *item* tidak sama dengan bentuk gambar berwarna hitam
6. Event "*OnMauseUp*", Action : *BackToPosition*
   - Memeriksa apakah item kembali ke posisi semula setelah melepas drag
7. Terminator "*Out*"
   - Mengakhiri proses jika menjawab salah

**Table 4. 7 Pengujian *Test Flow Diagram Item Match* Jika Menjawab Salah**

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Pengujian prosedur pada *gameplay item match* jika menjawab salah | Menghasilkan hasil yang sesuai pada kebutuhan dan fungsionalitas tanpa ada *bug* pada setiap prosedur jika jika menjawab salah pada *gameplay item match* | Menghasilkan hasil yang sesuai pada kebutuhan dan fungsionalitas tanpa ada *bug* pada setiap prosedur jika jika menjawab salah pada *gameplay item match* |

### 4.2.2 Pengujian Terhadap Pengguna (*Focus Testing*)

Pengujian kepada pengguna dilakukan untuk memastikan bahwa pembuatan *game* telah sesuai dengan tujuan yang ingin dicapai yaitu *game* yang menyenangkan dan interaktif yang dapat digunakan sebagai media pembelajaran baru untuk memperkenalkan bangun datar 2D. Pengujian dilakukan dengan menanyakan langsung kepada anak yang baru saja bermain *game* edukasi "Shape Adventure".

Pertanyaan diajukan kepada 5 anak dari TK RA Muslimat NU 16. Dan berikut hasil yang ditunjukan pada Tabel 4.8.

**Table 4. 8 Hasil Pengujian *Game* Terhadap Beberapa *Sample***

| No | Kriteria dan Tujuan | Yang Diharapkan | Anak ke-1 | Anak ke-2 | Anak ke-3 | Anak ke-4 | Anak ke-5 |
|---|---|---|---|---|---|---|---|
| 1 | Apakah permainan ini dapat menyenangkan jika di realisasikan dalam bentuk game digital? | Baik | Ya | Ya | Ya | Ya | Ya |
| 2 | Apakah permaianan ini memberikan gambaran tentang bangun datar 2D? | Ya | Ya | Ya | Ya | Ya | Ya |
| 3 | Apa strategi anda untuk menang? | kondisional | Menghapal bentuk bangun datar 2D | Mencoba terus dan terus | Berlatih menggunakan keyboard dan mouse | Berlatih menggunakan keyboard dan mouse | Berlatih menggunakan keyboard dan mouse |

**Table 4. 9 Hasil Pengujian *Game* Terhadap Beberapa *Sample (lanjutan)***

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4 | Apakah tujuan dari permainan ini sudah jelas pada setiap level? | Jelas | Jelas | Jelas | Jelas | Jelas | Jelas |
| 5 | Apakah ada sesuatu dalam permainan yang membuat anda bingung? | Tidak | Tidak | Tidak | Tidak | Tidak | Tidak |
| 8 | Apakah jenis gameplay ini dapat membuat pemain lebih mengingat dan dapat membentuk suatu bentuk baru dari berbagai macam jenis bentuk bangun 2D? | Ya | Ya | Ya | Ya | Ya | Ya |

# BAB 5 PEMBAHASAN

Pada tahap ini akan dibahas tentang hasil pengujian yang bertujuan untuk mendapatkan kesimpulan dari hasil pengujian *game* "Shape Adventure" yang sudah dilakukan. Proses pembahasan mengacu pada dasar teori sesuai dengan hasil pengujian yang didapatkan. Proses pembahasan yang dilakukan meliputi pembahasan hasil pengujian *basis path*, pembahasan pengujian *test flow diagram* dan pembahasan pengujian terhadap pengguna.

## 5.1 Pembahasan Hasil Pengujian *Basis Path*

Dari pengujian yang dilakukan terhadap beberapa metode program penting atau inti dengan metode pengujian *basis path* menghasilkan nilai kompleksitas siklomatis yang beragam. Dimulai dari metode menggerakkan *character* yang mendapatkan nilai kompleksitas siklomatis 6. Dari nilai tersebut maka ditentukan 6 buah basis set dari jalur independent untuk mendapatkan kasus uji. Kasus uji yang didapat yaitu menekan tombol "D" ketika *character* menghadap kanan, menekan tombol "D" ketika *character* menghadap kiri, menekan tombol "A" ketika *character* menghadap kiri, menekan tombol "A" ketika *character* menghadap kanan, menekan tombol "Space", dan menekan tombol "D" ketika *character* menghadap kanan sambil menekan tombol "Space". Dari kasus uji tersebut didapatkan hasil berupa *character* berjalan ke kanan ketika menekan tombol "D" saat *character* menghadap kanan, *character* berbalik menghadap kanan kemudian berjalan ke kanan ketika menekan tombol "D" saat *character* menghadap kiri, *character* berjalan ke kiri ketika menekan tombol "A" saat *character* menghadap kiri, *character* berbalik menghadap kiri kemudian berjalan ke kiri ketika menekan tombol "A" saat *character* menghadap kanan, *character* melompat ke atas ketika menekan tombol "Space" dan *character* akan melompat keatas serta bergerak ke kanan ketika menekan tombol "D" saat *character* menghadap kanan sambil menekan tombol "Space".

Metode mencocokan *item* mendapatkan nilai kompleksitas siklomatis 2. Dari nilai tersebut maka ditentukan 2 buah basis set dari jalur independent untuk mendapatkan kasus uji. Kasus uji yang didapat yaitu pengecekan item jika fix bernilai *false* dan pengecekan item jika fix bernilai *true*. Dari kasus uji tersebut didapatkan hasil berupa item kembali ke posisi semula jika fix bernilai *false* dan item menghilang serta gambar hitam menjadi berwarna jika fix bernilai *true*.

Metode menggeser *item* mendapatkan nilai kompleksitas siklomatis 2. Dari nilai tersebut maka ditentukan 2 buah basis set dari jalur independent untuk mendapatkan kasus uji. Kasus uji yang didapat yaitu pengecekan *item* jika di *drag* atau ada inisialisasi nilai pada transform.root.position saat meng*eklik* kiri *mouse* dan pengecekan *item* jika tidak di *drag* atau tidak ada inisialisasi nilai pada transform.root.position. Dari kasus uji tersebut didapatkan hasil berupa *item* bergerak sesuai pergerakan *mouse* saat *item* di *klik* kemudian di *drag* dan item tidak bergerak saat kita tidak meng*eklik item* serta tidak melakukan *drag* pada *item*.

Metode menang mendapatkan nilai kompleksitas siklomatis 3. Dari nilai tersebut maka ditentukan 3 buah basis set dari jalur independent untuk mendapatkan kasus uji. Kasus uji yang didapat yaitu pengecekan jumlah *item* yang benar jika kurang dari jumlah soal dan waktu lebih dari 0, pengecekan jumlah *item* yang benar kurang dari jumlah soal tetapi waktu menunjukan angka 0, dan pengecekan jumlah *item* benar sama dengan jumlah soal. Dari kasus uji tersebut didapatkan hasil berupa waktu akan berkurang terus sebanyak 1 detik sampai 0 jika jumlah *item* yang benar kurang dari soal dan waktu masih lebih dari 0, menampilkan *screen lose* jika jumlah *item* yang benar kurang dari soal tetapi waktu telah habis dan akan menampilkan *screen win* jika jumlah *item* yang benar sama dengan jumlah soal.

## 5.2 *Pembahasan Hasil Pengujian Test Flow Diagram*

Pengujian menggunakan metode test flow diagram ini bertujuan untuk memudahkan meninjau dan menganalisis desain yang akan diuji. Desain yang di uji dengan metode ini adalah ketika pemain mendapatkan item, pemain menjawab dengan benar dan pemain menjawab salah.

Saat pengujian dengan kasus pemain mendapatkan item dilakukan dengan cara membuat *character* berjalan mendekati *item* dan menyentuhnya. Saat dilakukan pengujian tersebut ketika character menyentuh item, item menghilang disertai dengan munculnya suara dan gambar item muncul di kotak sebelah pojok kiri atas dalam screen. Hasil tersebut telah sesuai dengan yang diharapkan saat pemain menjawab benar.

Pengujian dengan kasus pemain menjawab benar dilakukan dengan cara meng*eklik item* yang telah disediakan, kemudian melakukan *drag* kearah *item* yang memiliki bentuk serupa dan melepas *drag* pada saat *item* menyentuh gambar yang berbentuk sama dengan *item*. Saat dilakukan pengujian tersebut *item* bergerak sesuai dengan arah yang kita inginkan ketika kita melakukan *drag* pada mouse dan *item* menghilang ketika kita melepas *drag* saat item berada pada gambar yang berbentuk sama dengan *item*. Hasil tersebut telah sesuai dengan yang diharapkan saat pemain menjawab benar.

Pengujian dengan kasus menjawab salah dilakukan dengan cara yang hampir sama seperti menjawab benar, hanya saja yang membedakannya adalah ketika melepas drag *item*, *item* tidak menyentuh gambar yang bentuknya mirip dengan *item* yang di*drag*. Dan saat dilakukan pengujian, item kembali ke tempat semula saat pemain melepas *drag* dan item tidak menyetuh gambar yang cocok dengan *item* tersebut. Hasil ini telah sesuai dengan yang diharapkan saat pemain menjawab salah.

## 5.3 *Pembahasan Hasil Pengujian Terhadap Pengguna*

Saat pertama kali mengenalkan game "Shape Adventure" kepada anak-anak kelas TK B di TK RA Muslimat NU 16 Malang mendapatkan antusias yang tinggi dari anak-anak. Anak-anak begitu penasaran dengan *game* yang dikenalkan dan tidak sabar untuk mencoba memainkan *game*. Karena *game* ini hanya bisa dimainkan oleh satu *player* maka anak-anak banyak yang berebut untuk memainkannya. Untuk mengatasi hal tersebut digunakanlah system antrian. Setiap anak diberi waktu sekitar 5 menit untuk bermain *game* agar anak yang ingin bermain selanjutnya tidak terlalu lama menunggu.

Beberapa pertanyaan terkait mengenai *game* diberikan kepada anak yang pertama kali selesai mencoba memainkan *game* sebagai salah satu *sample* pengujian. Kemudian untuk selanjutnya, pengambilan sample dilakukan secara acak dari beberapa anak yang sudah mencoba permainan untuk diberikan pertanyaan. Dari beberapa pertanyaan yang dilontarkan beberapa anak ada yang menjawab berbeda. Contohnya saat diberi pertanyaan apa strategi anda untuk menang ada anak yang menjawab dengan mencoba terus, kemudian ada yang menjawab sering-sering menggunakan *keyboard* dan *mouse* karena belum terbiasa, ada yang menjawab latihan menggunakan *keyboard* dan *mouse*. Selebihnya beberapa pertanyaan yang dilontarkan mereka menjawabnya sama.

# BAB 6 PENUTUP

## 6.1 Kesimpulan

Berdasarkan hasil analisa, perancangan, implementasi dan pengujian yang dilakukan, maka dapat diambil kesimpulan sebagai berikut :

1. Untuk merancang *game* "Shape Adventure", sebagai sarana pembelajaran untuk anak pra sekolah dengan menggunakan MDA *framework* dimulai dengan perancangan untuk menentukan *elemen* formal dari *game* yang akan dibuat, membuat *prototype game* berdasarkan hasil dari perancangan agar bisa dimainkan, melakukan *play testing* dengan prototype untuk mengevaluasi apakah *rule* yang dibuat sudah sesuai tujuan dan menyenangkan, apabila belum maka akan kembali ke perancangan untuk menentukan *rule* kembali, membuat *prototype*, *play testing*, dan evaluasi.
2. *Game* "Shape Adventure" berhasil diimplementasikan kedalam bentuk *digital* dengan menggunakan *game engine* Unity dengan bahasa pemrograman C# dan dapat digunakan sebagai sarana media pembelajaran mengenal bangun datar 2D untuk anak pra sekolah.
3. Berdasarkan hasil pengujian *basis path testing* dengan menggunakan *whitebox testing,* didapatkan jumlah jalur pada logika setiap *method* dan prosedur telah sesuai dengan penghitungan *cyclomatic complexity* dan setiap jalur independent dengan hasil pengujian sesuai dengan hasil yang diharapkan.
4. Berdasarkan hasil pengujian *test flow diagram* menggunakan teknik jalur *minimum path generation,* didapatkan hasil pengujian sesuai dengan hasil yang diharapkan tanpa adanya *bug* dan tidak ditemukannya kesalahan fungsional.
5. Berdasarkan hasil pengujian terhadap pengguna (*Focus Testing*), beberapa pertanyaan yang diajukan mendapat tanggapan yang positif dari anak-anak TK yang memainkannya. Maka *game* Shape Adventure telah sesuai dengan tujuan pembuatannya yaitu dapat menjadi sarana pembelajaran yang menarik untuk anak TK dalam mengenal bangun datar 2D.

## 6.2 Saran

Saran untuk pengembangan *game* "Shape Adventure" lebih lanjut antara lain:

1. Dapat dilakukan pengembangan menjadi *multi-platform.*
2. Penambahan *level* baru dengan gambar *puzzle* yang lebih beragram.
3. Penambahan fitur-fitur lain seperti galeri yang dapat melihat hasil dari *puzzle* yang sudah diselesaikan.
4. Penambahan *character* perempuan agar bisa lebih menarik minat untuk anak perempuan.

# DAFTAR PUSTAKA


Hejlsberg, A., Torgersen, M., Wiltamuth, S. & Golde, P., 2011. *The C# Programming Language (Covering C# 4.0).* Addison-Wesley Professional.

Hunicke, R., LeBlanc, M. & Zubek, R., 2004. MDA: A Formal Approach to Game Design and Game Research. *the AAAI Workshop on Challenges in Game AI (Vol. 4).*

Kusni, 2008. *Geometri Dasar.* Semarang: Jurusan Matematika Universitas Negeri Semarang.

Martono, K. T., 2011. Perancangan Game Edukasi "Fish Identity" Dengan Menggunakan JavaTM. *Department of Computer System Engineering, Diponegoro University - Indonesia.*

Negoro, S. & Harahap, . B., 2003. *Ensiklopedia Matematika.* Jakarta: Ghalia Indonesia.

Pedersen, R. E., 2008. *Game Design Foundations, 2nd Edition.* Wordware Publishing, Inc.

Pressman, R. S., 2001. *Software Engineering: A Practitioner's Approach, 7th Edition.* Mc Graw Hill.

Roedavan, R., 2014. *UNITY Tutorial Game Engine.* Bandung: Informatika.

Rogers, S., 2010. *Level Up! The Guide to Great Video Game Design.* US: Willey.

Schereiber, I., 2009. *Game Design Concept.* US: Creative Commons Attribution 3.0.

Soetjiningsih, 1995. *Tumbuh Kembang Anak.* Jakarta: EGC.

Tarigan, D., 2006. *Pembelajaran Matematika Realistik.* Jakarta: Depdiknas.

Taurena D, M., 2008. *Analisis Ergonomi Dan Usulan Perbaikan Fasilitas Belajar Untuk Anak-Anak.*