BAB II
TINJAUAN PUSTAKA

2.1 Adab Islami

Adab secara bahasa artinya menerapkan akhlak mulia (Purnama, 2018). Dalam Fathul Bari, Ibnu Hajar menyebutkan:

وَالْأَدَبُ اسْتِعْمَالُ مَا يُحْمَدُ قَوْلًا وَفِعْلًا وَعَبَّرَ بَعْضُهُمْ عَنْهُ بِأَنَّهُ الْأَخْذُ بِمَكَارِمِ الْأَخْلَاقِ

"*Al-adab* artinya menerapkan segala yang dipuji oleh orang, baik berupa perkataan maupun perbuatan. Sebagian ulama juga mendefinsikan, adab adalah menerapkan akhlak-akhlak yang mulia." (Fathul Bari, 10/400)

Seorang muslim hendaknya selalu menjaga adab dan menghiasi dirinya dengan akhlak yang mulia dalam berinteraksi dengan siapapun. Akhlak yang mulia menjadi tolok ukur kebaikan seseorang. Rasulullah *shallallahu 'alaihi wa sallam* bersabda:

أَكْمَلُ الْمُؤْمِنِيْنَ إِيْمَانًا أَحْسَنُهُمْ خُلُقًا

"Orang mukmin yang paling sempurna imannya adalah yang terbaik akhlaknya." (HR. At-Tirmidzi no. 1162)

Secara umum, adab terbagi menjadi tiga macam. Pertama, adab kepada Allah *Subhanahu Wa Ta'ala.* Kedua, adab kepada Rasulullah *shallallahu 'alaihi wa sallam*. Ketiga, adab kepada sesama makhluk. Beradab dengan adab-adab Islam diantaranya: memberi salam, bermuka manis, makan dengan tangan kanan, minum dengan tangan kanan, membaca "bismillah" ketika mulai makan dan mulai minum, memuji kepada Allah membaca doa ketika selesai makan dan minum, memuji kepada Allah mengucapkan "alhamdulillah" ketika bersin, mendoakan orang yang bersin jika ia mengucapkan "alhamdulillah", menjenguk orang yang sakit, mengiringi jenazah untuk dishalati dan dimakamkan, adab-adab dalam syariat ketika masuk masjid, ketika masuk ke dalam rumah, ketika keluar dari masjid dan keluar dari rumah, ketika safar (melakukan perjalanan), ketika memakai pakaian, ketika melepaskannya, dan ketika memakai sandal, juga adab kepada orang tua, kepada kerabat, kepada tetangga, kepada orang yang lebih tua, kepada orang yang lebih muda, mendoakan bagi orang yang baru mendapatkan anak, mendoakan keberkahan bagi orang yang baru menikah, menghibur orang

yang terkena musibah dan adab-adab islamiyah lainnya.

2.2 *Game*

Game merupakan salah satu sarana hiburan yang banyak digemari banyak kalangan dari anak-anak, remaja, hingga dewasa. Bermain *game* juga dapat menjadi sarana pembelajaran atau edukasi. *Game* adalah suatu bentuk aktifitas bermain, bertingkah laku dalam konteks realita buatan, dimana partisipan mencoba untuk meraih minimal satu tujuan yang penting sesuai dengan keputusan *player* dengan bertindak sesuai dengan aturan (Adams, 2010).

Dalam Bahasa Indonesia, *game* diartikan sebagai permainan. Permainan merupakan bagian dari bermain dan bermain juga bagian dari permainan keduanya saling berhubungan. Permainan adalah kegiatan yang kompleks yang didalamnya terdapat peraturan, permainan dan budaya. Sebuah permainan adalah sebuah sistem dimana pemain terlibat dalam konflik buatan. Pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan. Dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan. *Game* bertujuan untuk menghibur. Biasanya *game* banyak disukai oleh anak–anak hingga orang dewasa. Terdapat macam-macam *game*, antara lain:

1. Aksi

   Genre ini merupakan jenis *game* yang paling populer. *Game* jenis ini membutuhkan kemampuan refleks pemain. Salah satu sub genre *action* yang popular adalah *First Person Shooter* (FPS). Pada *game* FPS diperlukan kecepatan berfikir. *Game* ini dibuat seolah-olah pemain yang berada dalam suasana tersebut.

2. Aksi Petualangan

   Genre ini memadukan *game play* aksi dan petualangan. Contohnya pemain diajak untuk menelusuri gua bawah tanah sambil mengalahkan musuh dan mencari artefak kuno atau menyeberangi sungai.

3. Simulasi, Konstruksi dan Manajemen

   Pemain dalam *game* ini diberi keleluasaan untuk membangun dan suatu proyek tertentu dengan bahan baku yang terbatas.

4. *Role Playing Games* (RPG)

   Dalam RPG, pemain dapat memilih satu karakter untuk dimainkan. Seiring dengan naiknya level *game*, karakter tersebut dapat berubah, bertambah kemampuannya, bertambah senjatanya atau bertambah hewan peliharaannya.

5. Strategi

   Genre strategi menitikberatkan pada kemampuan pada kemampuan berpikir dan organisasi. *Game* strategi dibedakan menjadi dua, yaitu *Turn Based Strategy* dan *Real Time Strategy*. *Real Time Strategy* mengharuskan pemain membuat keputusan dan secara bersamaan pihak lawan juga beraksi hingga menimbulkan serangkaian kejadian dalam waktu yang sebenarnya, sedangkan *Turn Based Strategy* pemain bergantian menjalankan taktiknya. Saat pemain mengambil langkah, pihak lawan menunggu, demikian juga sebaliknya.

6. Balapan

   Pemain dapat memilih kendaraan, lalu melaju di arena balap. Tujuannya yaitu mencapai garis *finish* tercepat.

7. Olahraga

   Genre ini membawa olahraga ke dalam sebuah komputer atau *console*. Biasanya *game play* dibuat semirip mungkin dengan kondisi olahraga yang sebenarnya.

8. *Puzzle*

   Genre *Puzzle* menyajikan teka-teki, menyamakan warna bola, perhitungan matematika, menyusun balok atau mengenal huruf dan gambar.

9. *Word Game* (Permainan Kata)

   Word Game sering dirancang untuk menguji kemampuan dengan bahasa atau untuk mengeksplorasi sifat-sifatnya. *Word Game* umumnya digunakan sebagai sumber hiburan, tetapi telah dibuktikan untuk melayani suatu tujuan pendidikan juga.

2.3 *Game* Edukasi

Menurut Pujiadi (2013), *game* yang memiliki konten pendidikan lebih dikenal dengan istilah *game* edukasi. *Game* edukasi ini bertujuan untuk menarik perhatian anak-anak dalam belajar sambil bermain dengan harapan para siswa

lebih mudah memahami materi pelajaran yang disajikan di sekolah. Jenis *game* ini sebenarnya lebih mengacu kepada isi dan tujuan *game*, bukan jenis yang sesungguhnya.

Menurut Edward (2009), *game* merupakan sebuah *tools* yang efektif untuk mengajar karena mengandung prinsip-prinsip pembelajaran dan teknik instruksional yang efektif digunakan dalam penguatan pada level-level yang sulit.

Game edukasi adalah permainan yang dirancang atau dibuat untuk memberikan pengajaran menambah pengetahuan penggunanya melalui suatu media unik dan menarik. *Game* jenis ini biasanya ditujukan untuk anak-anak, maka permainan desain sangat diperlukan (Handriyantini, 2009).

Berdasarkan uraian di atas maka dapat disimpulkan *game* edukasi adalah salah satu bentuk *game* yang dapat berguna untuk menunjang proses belajar-mengajar secara lebih menyenangkan dan lebih kreatif dan digunakan untuk memberikan pengajaran atau menambah pengetahuan penggunanya melalui suatu media yang menarik.

2.4 *Game* Sebagai Media Pembelajaran

Bermain merupakan sesuatu yang tidak terpisahkan dalam kehidupan anak. Bermain *game* dipandang sebagian masyarakat sebagai sesuatu yang menyebabkan malas belajar bagi anak, tetapi disamping itu bermain *game* mempunyai manfaat yang cukup banyak asalkan tepat pada sasaran. Menurut Kemp dan Dayton (1985), manfaat *game* sebagai media pembelajaran adalah sebagai berikut.

1. Penyeragaman penyampaian materi pelajaran

   Dengan menggunakan media *game* maka orang tua atau pendidik dapat menyeragamkan materi pembelajaran yang akan disampaikan kepada anak.

2. Proses pembelajaran lebih menarik

   Dengan adanya media *game* maka pembelajaran yang terjadi tidaklah menjadi membosankan. Anak dapat berinteraksi dengan apa yang dimainkan, baik dengan audio, video maupun gerak.

3. Proses belajar anak menjadi lebih interaktif

   Dengan adanya media *game*, proses belajar lebih interaktif. Anak akan lebih terangsang untuk melakukan proses pembelajaran. Misalnya dengan adanya soal yang harus dikerjakan untuk bisa meneruskan misi permainan.

4. Jumlah waktu belajar mengajar dapat dikurangi

   Dengan *game*, orang tua maupun pendidik akan lebih efisisen untuk melakukan pembelajaran. Pendidik hanya mengarahkan dan menjelaskan secara singkat media *game* yang sudah berisi tentang materi atau bahan ajar.

5. Kualitas belajar anak dapat ditingkatkan

   Dengan media *game*, kualitas pembelajaran akan lebih baik, karena tidak hanya mengerti materi belajar, anak dituntut untuk lebih kreatif dan teliti untuk menyelesaikan misi yang terdapat pada permainan tersebut.

6. Proses belajar dapat terjadi dimana saja dan kapan saja

   Dengan media *game*, proses pembelajaran tidak hanya dilakukan di sekolah, tapi bisa dilakukan dimana saja dan kapan saja.

2.5 Construct 2

Construct 2 adalah sebuah *tool* berbasis *Hyper Text Markup Language* (HTML) 5 untuk menciptakan sebuah *game*. HTML 5 merupakan bahasa markup untuk penataan dan penyajian konten untuk *World Wide Web* dan merupakan teknologi inti dari jaringan internet yang pada awalnya diusulkan oleh Opera Software (Gullen, 2011).

Construct 2 berbeda dengan *tools* lain yang mengharuskan pemrogram menuliskan baris demi baris agar tercipta sebuah objek. Hal ini karena Construct 2 sudah berbasis objek sehingga sangat mudah dalam membuat objek-objek dan mengatur atribut-atribut dari objek tersebut. Construct 2 juga memiliki fitur-fitur yang mudah digunakan dan dimengerti oleh pemrogram pemula.

Construct 2 dikembangkan dengan tujuan memudahkan *non-programmer* yang ingin menciptakan *game* secara *drag* and *drop* dengan editor visual dan berbasis sistem logika perilaku. Editor visual adalah tempat dimana objek-objek diletakkan atau dibuat. Adapun pengaturan logika perilaku masing-masing objek yang dinamakan *event* dan dituliskan dalam *event sheet*. *Event*

dalam Construct 2 merupakan kumpulan dari *conditions* dan *actions*. *Conditions* menjelaskan kondisi objek yang ada, sedangkan *actions* adalah aksi yang menggerakkan objek-objek tersebut (Gullen, 2011). Construct 2 dirancang untuk pengembangan *game* berbasis 2D.

Construct 2 juga menyediakan bermacam-macam visual *effect* yang menggunakan *engine* WebGL dan *plugin* serta *behaviour* yang dapat membantu para pengembang dalam menciptakan aplikasi yang menarik dan interaktif. Pemanggilan fungsi-fungsi yang ada di dalam Construct 2 hanya dengan menggunakan pengaturan *event* yang telah disediakan (Gullen dan Gullen, 2011).

Gambar 2.1 Tampilan *start page* construct 2

Bagian ruang kerja dalam Construct 2 dibedakan sebagai berikut (Scirra, 2014):

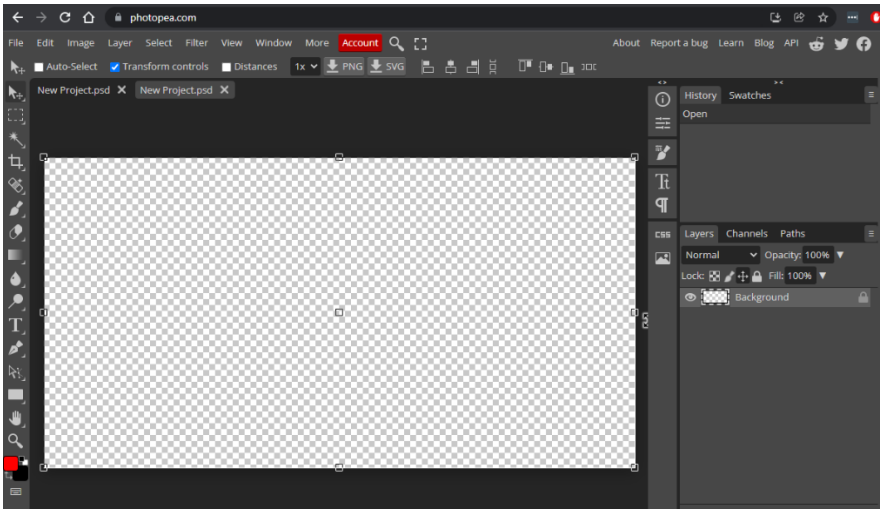
1. Area Kerja Construct 2, untuk menggambarkan berbagai objek yang dibuat, seperti objek *sprite*, objek *background*, dan objek lainnya.
2. Menu *Properties* Construct 2, untuk mengatur kebutuhan objek yang dibuat, seperti warna *layout*, ukuran objek *sprite*, dan lainnya.
3. Menu *Projects* dan *Layers*, *projects* untuk memilih *project* yang akan dikerjakan sedangkan *layer* untuk membuat beberapa *layer* dalam suatu *layout* kerja.
4. Menu *Library*, yaitu tempat untuk menyimpan kumpulan dari objek-objek yang telah dibuat.
5. *Event sheet*, yaitu area kerja Construct 2 untuk menulis *event-event* yang akan menggerakkan objek-objek yang telah dibuat.

2.6 Photopea

Photopea adalah suatu *website* editor gambar tingkat lanjut yang dapat mengolah grafik *raster* dan vektor. Photopea dapat digunakan untuk tugas-tugas sederhana, seperti mengubah ukuran gambar, serta tugas-tugas kompleks, seperti merancang halaman *web*, membuat ilustrasi, memproses foto dan banyak lagi.

Saat ini, Photopea berjalan dengan ruang warna sRGB (ruang warna dasar untuk *web*) dengan kedalaman warna 8-bit. Semua *file* yang diekspor juga menggunakan sRGB.

Editor Photopea bekerja di *browser web*. Hal ini dapat dimulai dengan pergi ke situs www.Photopea.com. Photopea dapat berjalan di perangkat apapun (*desktop*, laptop, tablet, ponsel, atau komputer lain manapun), tetapi untuk kenyamanan terbaik, disarankan untuk memiliki layar besar, perangkat petunjuk yang tepat (*mouse* atau *stylus*) dan *keyboard*.



Gambar 2.2 Tampilan *workspace* photopea

Ruang kerja Photopea sangat mirip dengan editor gambar lainnya, terdiri dari *Toolbar* di sebelah kiri, *Sidebar* di sebelah kanan, area kerja di bagian tengah dan *Top Menu* di bagian atas.

Photopea menggunakan format PSD sebagai format utama untuk menyimpan dokumen gambar dengan informasi tambahan. Hal itu dirancang untuk digunakan dalam Adobe Photoshop dan menjadi sangat populer saat itu. Semua *file* yang dibuka di Photopea (seperti PNG, JPG, Sketch) dikonversi ke PSD (jika belum menjadi PSD). Setelah selesai mengedit, hasilnya dapat disimpan

dari PSD ke format lain (Photopea, 2017).

2.7 Apache Cordova

Apache Cordova adalah sebuah *framework* untuk membangun aplikasi *mobile native* menggunakan HTML, CSS dan JavaScript. *Native mobile application* yang didukung antara lain Android, iOS, Windows Phone dan Blackberry.

Apache Cordova berisi sekumpulan API (*Application Programming Interface*) untuk mengakses *device* dari perangkat *mobile*. *Device* tersebut antara lain kamera, GPS (*Global Positioning System*), *storage* dan lain-lain. Dengan menunggunakan UI (*User Interface*) *framework* seperti jQuery Mobile, Dojo Mobile atau Sencha Touch, maka API tersebut dapat diakses. Dengan kata lain, kita dapat membangun aplikasi hanya menggunakan HTML, CSS dan Javascript.

Dengan menggunakan API dari Cordova, kita tidak perlu membangun aplikasi menggunakan *native code* seperti Java, Objective-C dan lainnya. Kita hanya menggunakan teknologi *web* dengan bahasa pemrograman *web* dan di-*install* pada perangkat *mobile* yang bersangkutan. Dikarenakan menggunakan Javascript, aplikasi yang kita bangun pada suatu *platform mobile* dapat digunakan di *platform* lain dengan sedikit atau tanpa perubahan.

Aplikasi yang dihasilkan dari Cordova dikemas dalam aplikasi menggunakan SDK masing-masing *platform* dan dapat diterapkan ke dalam *platform* lain menggunakan SDK *platform* tersebut.

Apache Cordova cocok digunakan untuk:

1. Seorang *developer* perangkat *mobile* yang ingin memperluas suatu aplikasi ke lebih dari satu *platform*, cukup membuat sekali *coding* tanpa harus melakukan implementasi ulang pada setiap *platform.*
2. Seorang *web developer* yang ingin menerapkan aplikasi *web* yang perlu dijalankan pada perangkat *mobile* untuk mengakses aplikasi yang dibuatnya.
3. Seorang *developer* perangkat *mobile* yang tertarik menggabungkan komponen aplikasi *native* dengan *WebView* (Cordova, 2015).

2.8 Android

Android, Inc. berdiri di kota Palo Alto, salah satu kota terkenal di

California (USA), tepatnya pada bulan Oktober tahun 2003. Pendirinya terdiri dari tiga orang yaitu Andy Rubin, Rich Miner, dan Chris White. Mereka adalah para ahli dalam pengembangan aplikasi. *Operating System* (OS) ini dikembangkan secara diam-diam meskipun dibuat oleh orang yang ahli di bidang pengembangan aplikasi. Pada tanggal 17 Agustus 2005, Google membeli OS ini secara penuh dan menjadikan salah satu produk unggulannya. Setelah penantian cukup panjang, akhirnya perusahaan yang berbasis di California ini mengumumkan pada 5 November 2007 bahwa mereka sedang merancang *open source* OS baru bernama Android yang nantinya akan menyaingi Symbian, Mac, Microsoft dan lain-lain.

Android merupakan sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan *platform* terbuka bagi para pengembang aplikasi mereka sendiri untuk digunakan oleh bermacam piranti bergerak. Kemudian untuk mengembangkan Android, dibentuklah *open handset alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia (Huda, 2016).

Para *developer* bisa mengembangkan sendiri aplikasi sesuai dengan keinginan mereka sendiri dengan menggunakan *Software Development Kit* (SDK) yang Google telah menditribusikannya untuk umum. Android termasuk OS yang cepat berevolusi karena dengan semakin bertambahnya aplikasi yang di sediakan oleh Google sendiri maupun oleh *developer* sendiri yang nantinya ini bisa di unduh melalui Google Play Store.

Sejak pertama kali Android dirilis, sudah banyak sekali versi-versi dari Android dibuat. Hal ini membuktikan bahwa OS ini berkembang begitu pesat. Sampai saat ini paling tidak sudah ada 17 versi Android yang beredar. Namun, sebelum ada kesembilan belas versi itu sebenarnya telah muncul Android Alpha dan Beta yaitu antara tahun 2007 hingga 2008. Perlu diketahui, nama versi Android berdasarkan nama makanan atau hidangan penutup. Hal ini mungkin sengaja dibuat agar mudah untuk di ingat. Berikut ini urutan versi Android dari yang paling awal hingga yang terbaru di tahun 2021:

1. Android 1.0 (Alpha)
2. Android 1.1 (Beta)
3. Android 1.5 (Cupcake)

4. Android 1.6 (Donut)
5. Android 2.0 – 2.1 (Eclair)
6. Android 2.2 (Frozen Yoghurt – Froyo)
7. Android 2.3 (Gingerbread)
8. Android 3.0 – 3.2 (Honeycomb)
9. Android 4.0 (Ice Cream Sandwich)
10. Android 4.1 – 4.3 (Jelly Bean)
11. Android 4.4 (KitKat)
12. Android 5.0 (Lollipop)
13. Android 6.0 (Marshmallow)
14. Android 7.0 – 7.1 (Nougat)
15. Android 8.0 – 8.1 (Oreo)
16. Android 9 (Pie)
17. Android 10 (Android Q)
18. Android 11
19. Android 12 (Snow Cone)

2.9 Algoritma *Fisher Yates Shuffle*

Fisher Yates Shuffle (dinamai berdasarkan penemunya, Ronald Fisher dan Frank Yates), juga dikenal sebagai *Knuth Shuffle* (diambil dari nama Donald Knuth) adalah sebuah algoritma untuk menghasilkan permutasi acak dari suatu himpunan terhingga, dengan kata lain untuk mengacak suatu himpunan tersebut.

Pemakaian *Fisher Yates Shuffle* bisa melalui dua cara yaitu: *Original Method* dan *Modern Method*. Menurut Pavel (2011), *Original Method* dipublikasikan pada tahun 1938, pada metode ini dilakukan dengan cara penarikan secara berulang dari unsur daftar masukan kemudian menuliskannya ke daftar keluaran kedua. Pendekatan ini dilakukan oleh manusia dengan secarik kertas dan sebuah pensil.

Pada *Modern Method* dijabarkan untuk penggunaan komputerisasi yang dikenalkan oleh Richard Durstenfield pada tahun 1964. *Modern Method* dikenalkan karena lebih optimal dibandingkan dengan *Original Method*. Algoritma yang modern berbeda dari yang sebelumnya, sangat komputasi dan

matematis. Prosesnya angka terakhir akan dipindahkan ke angka yang ditarik keluar dan mengubah angka yang ditarik keluar menjadi angka akhir yang tidak ditarik lagi untuk setiap kali penarikan dan berlanjut untuk iterasi berikutnya. Hal ini dilakukan dalam O (1) waktu dan ruang. Dengan demikian, waktu dan ruang kompleksitas algoritmanya O (n) yang optimal (O'Connor, 2014).

Berikut adalah metode modern *Fisher Yates Shuffle* yang digunakan untuk menghasilkan suatu permutasi acak angka 1 sampai n.

1. Tuliskan angka 1 sampai n.
2. Pilih sebuah angka acak k diantara 1 sampai dengan jumlah angka yang belum dianggap teracak (dicoret).
3. Hitung dari bawah, coret angka k yang belum dicoret dan tuliskan angka tersebut di lain tempat.
4. Ulangi langkah 2 dan langkah 3 sampai semua angka sudah tercoret.
5. Urutan angka yang dituliskan pada langkah 3 adalah permutasi acak dari angka awal.

Tabel 2.1 Contoh Pengerjaan Algoritma *Fisher Yates Shuffle*

Range	***Roll***	***Scratch***	***Result***
		12345678	
1-8	5	1234678	5
1-7	3	124678	35
1-6	4	12678	435
1-5	8	1267	8435
1-4	2	167	28435
1-3	7	16	728435
1-2	1	6	1728435
Hasil Pengacakan			61728435

Menurut Singh (2014), penggunaan algoritma *Fisher Yates Shuffle* yang modern oleh Richard Durstenfeld dapat mengurangi kompleksitas algoritma menjadi O (n), dibandingkan dengan mengacak menggunakan metode yang lain seperti menggunakan *sorting* yang sangat tidak efisien karena adanya *loop* bersarang.

Algoritma *Fisher Yates Shuffle* dipilih karena algoritma ini merupakan metode pengacakan yang lebih baik atau dapat dikatakan sesuai untuk pengacakan angka, dengan waktu eksekusi yang cepat serta tidak memerlukan waktu yang

lama untuk melakukan suatu pengacakan. Pengacakan suatu hal yang sangat penting dalam pembuatan banyak aplikasi. Meskipun terlihat mudah namun pada dasarnya jika tidak dilakukan dengan baik maka pengacakan itu dapat berdampak buruk untuk suatu aplikasi. Dalam hal ini pengacakan menggunakan algoritma *Fisher Yates Shuffle* dapat dijadikan referensi untuk diterapkan dalam sebuah aplikasi yang menggunakan metode pengacakan (Bendersky, 2010).

2.10 Algoritma *A Star*

A Star (A*) merupakan algoritma yang pertama kali dikembangkan oleh Nils Nilsson pada 1964 berdasarkan algoritma Dijkstra. Saat itu, algoritma ini dinamakan algoritma A1. Pada 1967, Bertram Raphael mengembangkan lebih jauh algoritma ini dan menyebutnya A2, namun tidak dapat membuktikan keunggulannya dibandingkan algoritma sebelumnya. Kemudian pada 1968 Peter E. menunjukkan bukti keoptimalan algoritma A2 dibandingkan dengan A1. Kemudian algoritma A2 dinyaatakan sebagai algoritma paling optimal untuk kasus tersebut, dan diganti namanya menjadi A*. Berdasarkan waktu, algoritma ini lebih baik daripada algoritma Dijkstra dengan pencarian heuristik.

A Star memiliki 2 fungsi utama dalam menentukan solusi terbaik. Fungsi pertama disebut sebagai g(x) merupakan fungsi yang digunakan untuk menghitung total *cost* yang dibutuhkan dari *node* awal menuju *node* tertentu. Fungsi kedua yang biasa disebut sebagai h(x) merupakan fungsi perkiraan total *cost* yang diperkirakan dari suatu *node* ke *node* akhir (Wafiqurrahman, 2015).

$$G(n) = \sqrt{Xn^2 + Yn^2}$$

$$H(n) = |X(target) - X(n)| + |Y(target) - Y(n)|$$

$$F(n) = G(n) + H(n)$$

Prinsip algoritma ini adalah mencari jalur terpendek dari sebuah simpul awal menuju simpul tujuan dengan memperhatikan harga (F) terkecil. Diawali dengan menempatkan A pada *starting point*, kemudian memasukkan seluruh simpul yang bertetangga dan tidak memiliki atribut rintangan dengan A ke dalam *open list* (tempat menyimpan data simpul yang mungkin diakses dari starting *point* maupun simpul yang sedang dijalankan). Kemudian mencari nilai F terkecil dari simpul- simpul dalam *open list* tersebut. Kemudian memindahkan A kesimpul

yang memiliki nilai F terkecil. Simpul sebelum A disimpan sebagai *parent* dari A dan dimasukkan kedalam *closed list* (tempat menyimpan data simpul sebelum A yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan).

Jika terdapat simpul lain yang bertetangga dengan A (yang sudah berpindah) namun belum termasuk kedalam anggota *open list*, maka masukkan simpul-simpul tersebut kedalam *open list*. Setelah itu, bandingkan nilai G yang ada dengan nilai G sebelumnya (pada langkah awal, tidak perlu dilakukan perbandingan nilai G). Jika nilai G sebelumnya lebih kecil maka A kembali ke posisi awal. Simpul yang pernah dicoba dimasukkan ke dalam *close list*. Hal tersebut dilakukan berulang-ulang hingga terdapat solusi atau tidak ada simpul lain yang berada dalam *open list* (Wafiqurrahman, 2015).

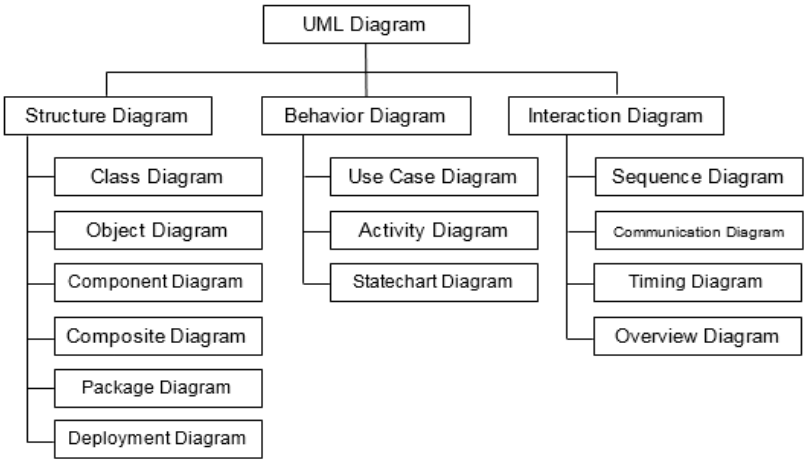
2.11 *Unified Modeling Language* (UML)

Sukamto dan Shalahuddin (2013) menjelaskan bahwa UML (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain serta menggambarkan arsitektur dalam pemrograman berorientasi objek.

UML menyediakan serangkaian gambar dan diagram yang sangat baik. Beberapa diagram memfokuskan diri pada ketangguhan teori *object-oriented* dan sebagian lagi memfokuskan pada detail rancangan dan konstruksi. Semua dimaksudkan sebagai sarana komunikasi antar *team programmer* maupun dengan pengguna.

Widodo dan Herlawati (2011) menjelaskan tentang kegunaan UML sebagai berikut.

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabarkan sistem secara rinci untuk menganalisis dan mencari apa yang diperlukan sistem.
4. Mendokumentasikan sistem yang ada, proses-proses dan organisasinya.



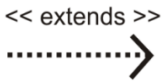
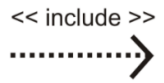
Gambar 2.3 Jenis kategori diagram UML

2.11.1 *Use Case Diagram*

Sukamto dan Shalahuddin (2013) menjelaskan *Use Case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem. *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Adapun simbol-simbol yang digunakan dalam *Use Case Diagram* adalah sebagai berikut:

Tabel 2.2 Simbol-Simbol *Use Case Diagram*

No.	Simbol	Nama	Deskripsi
1		*Actor*	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat. Biasanya nama aktor dinyatakan dengan menggunakan kata benda.
2		*Use Case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. Biasanya nama *Use Case* dinyatakan dengan menggunakan kata kerja.
3		*Association*	Komunikasi antara aktor dan Use

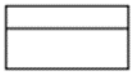
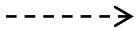
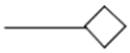
No.	Simbol	Nama	Deskripsi
			Case yang berpartisipasi pada diagram *Use Case* atau *Use Case* yang memiliki interaksi dengan aktor.
4		*Extend*	Relasi *Use Case* tambahan pada sebuah *Use Case*, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek. Arah panah mengarah pada *Use Case* yang ditambahkan.
5		*Generalization*	Hubungan generalisasi dan spesialisasi antara dua buah *Use Case* dimana fungsi *Use Case* turunan memiliki fungsi *Use Case* yang menjadi generalisasinya. Arah panah mengarah pada *Use Case* yang menjadi generalisasinya.
6		*Include*	Relasi tambahan pada sebuah *Use Case* dimana *Use Case* yang ditambahkan membutuhkan *Use Case* tambahan ini untuk menjalankan fungsinya. Arah panah mengarah pada *Use Case* yang ditambahkan dan *Use Case* ini selalu dijalankan bersamaan dengan *Use Case* tambahan.

2.11.2 *Class Diagram*

Sukamto dan Shalahuddin (2013) menjelaskan *Class Diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class Diagram* dibuat agar pembuat program

atau *programmer* membuat kelas-kelas sesuai rancangan di dalam *Class Diagram* agar antara dokumentasi perancangan dan perangkat lunak sinkron. Adapun simbol-simbol yang digunakan dalam *Class Diagram* adalah sebagai berikut:

Tabel 2.3 Simbol-Simbol *Class Diagram*

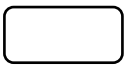
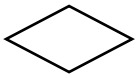
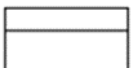
No.	Simbol	Nama	Deskripsi
1		*Class*	Kelas pada stuktur sistem.
2		*Interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
3		*Associaton*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
4		*Directed Association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.
5		*Generalization*	Relasi antar kelas dengan makna generalisasi-spesialisasi.
6		*Dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas.
7		*Aggregation*	Relasi antar kelas dengan makna semua bagian (*whole-part*).

2.11.3 *Activity Diagram*

Sukamto dan Shalahuddin (2013) menjelaskan *Activity Diagram* menggambarkan *workflow* atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan adalah bahwa *Activity Diagram* menggambarkan aktivitas yang dapat dilakukan oleh sistem bukan apa yang dilakukan aktor. *Activity Diagram* dapat menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang, bagaimana aliran tersebut berawal, *decision* yang mungkin terjadi dan akhir dari proses yang terjadi.

Adapun simbol-simbol yang digunakan dalam *Activity Diagram* adalah sebagai berikut:

Tabel 2.4 Simbol-Simbol *Activity Diagram*

No.	Simbol	Nama	Deskripsi
1		*Initial Node*	Status awal aktivitas sistem. Sebuah *Activity Diagram* memiliki sebuah status awal.
2		*Activity*	Aktivitas yang dilakukan sistem, biasanya diawali dengan kata kerja.
3		*Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4		*Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5		*Final Node*	Status akhir yang dilakukan sebuah sistem. Sebuah *Activity Diagram* memiliki sebuah status akhir.
6		*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.12 *Storyboard*

Menurut Vaughan (2011), *storyboard* adalah gambaran arsitektur yang digunakan pada proyek multimedia untuk membantu dalam hal memvisualisasikan mengenai informasi arsitektur proyek tersebut. *Storyboard* dapat menggambarkan proyek dengan detail menggunakan tulisan dan sketsa gambar yang digunakan untuk pilihan layar gambar, suara, dan navigasi setiap layarnya. Selain itu di dalam storyboard juga dapat mengatur warna dan corak, isi teks, atribut, font dan bentuk tombol, *style*, tanggapan serta perubahan suara.

Sebelum membuat *storyboard*, disarankan untuk membuat cakupan *storyboard* terlebih dahulu dalam bentuk rincian naskah yang kemudian akan dituangkan detail grafik dan visual untuk mempertegas dan memperjelas tema.

Metode yang akan digunakan tergantung pada apakah orang yang sama akan mengerjakan semuanya atau apakah implementasi akan ditugaskan dan dikerjakan oleh tim baru yang akan membutuhkan spesifikasi secara detail seperti *storyboard* dan sketsa. Disamping itu, semakin banyak rencana yang ada dikertas, semakin baik dan mudah dalam mengerjakan suatu proyek.

Storyboard memiliki banyak bentuk yang tergantung pada pembuatnya. Beberapa bentuk *storyboard* yang paling umum digunakan, yaitu:

1. Bentuk kasar yang berfungsi untuk menjelaskan produk secara abstrak.
2. *Concept storyboard,* yaitu *storyboard* yang penuh dengan warna dan *full atmosphere.*
3. *Color storyboard*, yaitu memetakan gambaran dengan penuh warna, bahkan terkesan lebih mengutamakan warna daripada urutan animasi.
4. *Animation storyboard,* yang berfungsi untuk menggambarkan secara detail animasi itu sendiri.
5. *Presentation storyboard*, yaitu bentuk storyboard yang di rancang dengan rapi dan sangat terencana, berfungsi untuk menjual suatu ide kepada klien.

2.13 Pengujian *Black Box*

Pengujian *Black Box* dilakukan untuk menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi- fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian *Black Box* dilakukan selama tahap akhir pengujian. Hal ini disebabkan karena pengujian *Black Box* memperhatikan struktur kontrol perangkat lunak. Pengujian *Black Box* berusaha untuk menemukan kesalahan dalam kategori berikut:

1. Fungsi yang tidak benar atau fungsi yang hilang,
2. Kesalahan antarmuka,
3. Kesalahan dalam struktur data atau akses *database* eksternal,
4. Kesalahan perilaku (*behavior*) atau kesalahan kinerja, dan
5. Inisialisasi dan pemutusan kesalahan.

Meskipun dirancang untuk mengungkap kesalahan, pengujian *Black Box* digunakan untuk memperlihatkan bahwa fungsi-fungsi perangkat lunak dapat beroperasi, bahwa *input* diterima dengan baik, *output* dihasilkan dengan tepat dan integritas informasi eksternal (seperti *file* data) dipelihara. Adapun Teknik pengujian *Black Box* adalah sebagai berikut:

1. *Boundary Value Analysis* (BVA) adalah banyaknya kesalahan terjadi pada masukan (*input*). *Boundary Value Analysis* mengijinkan untuk menyeleksi kasus uji yang menguji batasan nilai *input*. *Boundary Value Analysis* adalah teknik desain proses yang melengkapi *equivalence partitioning.* Teknik *Boundary Value Analysis* menyediakan beberapa pedoman dalam praktiknya. Pengaplikasian pedoman tersebut dapat mendeteksi kemungkinan kesalahan lebih besar.
2. *Comparison Testing* dilakukan jika reliabilitas suatu perangkat lunak sangat kritis. *Comparison Testing* dilakukan dengan mengembangkan perangkat lunak dalam versi independen yang berbeda. Setiap versi perangkat lunak diuji dengan data yang sama untuk memastikan semua versi menghasilkan keluaran yang sama.
3. *Sample Testing* adalah pengujian yang melibatkan beberapa nilai yang terpilih dari sebuah kelas ekuivalen. *Sample Testing* mengintegrasikan nilai pada kasus uji dan nilai-nilai yang terpilih, bisa dipilih dengan urutan tertentu atau interval tertentu.
4. *Robustness Testing* adalah pengujian dengan data *input* dipilih diluar spesifikasi yang telah didefiniskan. Tujuan dari pengujian ini adalah membuktikan bahwa tidak ada kesalahan jika masukan tidak valid.
5. *Behavior Testing* dilakukan untuk menguji perilaku sistem apakah sudah sesuai dengan permintaan atau polanya. *Behavior Testing* merupakan pengujian dengan hasil uji yang tidak dapat dievaluasi jika hanya melakukan pengujian sekali, tapi dapat dievaluasi jika pengujian dilakukan beberapa kali, misalnya pada pengujian struktur data.
6. *Requirement Testing* adalah spesifikasi kebutuhan yang terasosiasi dengan perangkat lunak (*input, output,* fungsi, performasi) dan diidentifikasikan pada tahap spesifikasi kebutuhan dan desain. *Requirement Testing* melibatkan

pembuatan kasus uji untuk setiap spesifikasi kebutuhan yang terkait dengan program,

7. *Perfomance Testing* (pengujian kinerja sistem) menguji *software* dari sistem/orientasi kepada *hardware*. Selain itu, mengevaluasi kemampuan program untuk beroperasi dengan benar yang dipandang dari sisi acuan kebutuhan, misalnya aliran data, ukuran pemakaian memori, kecepatan eksekusi dan lain-lain. *Performance Testing* digunakan untuk mencari tahu beban kerja atau kondisi konfigurasi program dan menguji batasan lingkungan program.
8. *Endurance Testing* (pengujian daya tahan sistem) adalah melibatkan kasus uji yang diulang-ulang dengan jumlah tertentu dengan tujuan mengevaluasi program apakah sesuai dengan spesifikasi kebutuhan.
9. *Equivalence Partitioning* adalah membagi input menjadi kelas-kelas data yang dapat digunakan untuk menggenerasikan kasus uji dengan tujuan untuk menemukan kelas-kelas kesalahan. Selain itu, *Equivalence Partitioning* berdasarkan pada kesamaan kelas-kelas kondisi *input*. Sebuah kelas yang ekuivalen merepresentasikan kumpulan status/kondisi yang valid atau tidak valid. Sebuah kondisi input dapat berupa nilai numerik yang spesifik, rentang nilai, kumpulan nilai yang berkaitan atau kondisi Boolean.
10. *Cause-Effect Relationship Testing* (pengujian sebab-akibat) adalah teknik yang merupakan suplemen dari *Equivalence Testing* dengan menyediakan cara untuk memilih kombinasi data input dan melibatkan kondisi input (*cause*) dan kondisi output (*effect*) untuk mencegah pendifinisian kasus uji yang terlalu banyak.
11. *User Acceptance Testing* (UAT) adalah pengujian terakhir sebelum sistem dipakai oleh *user* yang melibatkan pengujian dengan data dari pengguna sistem. *User Acceptance Test* adalah uji terima perangkat lunak yang dilakukan pengguna perangkat lunak tersebut. Tujuan pengujian ini adalah untuk menguji apakah sistem sudah sesuai dengan apa yang tertuang dalam spesifikasi fungsional sistem (*validation*).

2.14 Skala *Likert*

2.14.1 Pengertian Skala *Likert*

Skala *Likert* merupakan metode pengukuran yang digunakan untuk mengukur sikap, pendapat dan persepsi sesorang atau kelompok orang tentang fenomena sosial (Sugiyono, 2012).

2.14.2 Penentuan Skor Jawaban

Skor jawaban merupakan nilai jawaban yang akan diberikan oleh responden. Hal pertama yang harus kita lakukan adalah menentukan skor dari tiap jawaban yang akan diberikan. Contohnya, sikap yang akan kita pakai yaitu ”setuju”. Selanjutnya, kita menentukan banyaknya jawaban pada tiap pernyataan yang akan kita berikan. Misalnya 5 skala, berarti sangat tidak setuju, kurang setuju, cukup setuju, setuju dan sangat setuju. Jika pertanyaan yang diberikan bersifat susah untuk diberikan jawaban, otomatis responden cenderung statis. Oleh karena itu, kita dapat memberikan pilihan jawaban yang banyak, misal 7 atau 9 jawaban dari setiap pertanyaan. Hal ini bertujuan agar responden dapat memberikan penilaian sesuai dengan kriteria mereka berdasarkan pilihan yang ada (Sugiyono, 2012). Skala nilai dapat dilihat pada **Tabel 2.5**.

Tabel 2.5 Skala Nilai *Likert*

Skala Jawaban	Nilai
Sangat Tidak Setuju/Suka/Bagus	1
Kurang Setuju/Suka/Bagus	2
Cukup Setuju/Suka/Bagus	3
Setuju/Suka/Bagus	4
Sangat Setuju/Suka/Bagus	5

2.14.3 Skor Ideal

Skor ideal merupakan skor yang digunakan untuk menghitung skor untuk menemukan *rating scale* dan jumlah seluruh jawaban. Untuk menghitung jumlah skor ideal (kriterium) dari seluruh item digunakan rumus berikut, yaitu: skor kriterium = nilai skala x jumlah responden. Seandainya skor tertinggi adalah 5 dan jumlah responden 20, maka dapat dirumuskan menjadi:

Tabel 2.6 Rumus Skala *Likert*

Rumus	Skala
5 × 20 = 100	Sangat Baik
4 × 20 = 80	Baik
3 × 20 = 60	Cukup Baik
2 × 20 = 40	Kurang Baik
1 × 2 = 20	Sangat Kurang Baik

Selanjutnya semua jawaban responden dijumlahkan dan dimasukkan ke dalam *rating scale* dan ditentukan daerah jawabannya.

2.14.4 *Rating Scale* Skala *Likert*

Rating scale berfungsi untuk mengetahui hasil data angket (kuesioner) dan wawancara secara umum dan keseluruhan yang didapat dari penilaian angket (kuesioner) dan wawancara dengan ketentuan seperti pada **Tabel 2.7**.

Tabel 2.7 Hasil Penelitian Skala *Likert*

Rumus	Skala
81 – 100	Sangat Baik
61 – 80	Baik
41 – 60	Cukup Baik
21 – 40	Kurang Baik
0 – 20	Sangat Kurang Baik

2.14.5 Persentase Persetujuan

Untuk mengetahui jumlah jawaban dari para responden melalui persentase yaitu digunakan rumus sebagai berikut: p = f / n x 100%. Keterangan dari rumus tersebut adalah p: persentase; f: frekuensi dan n: jumlah skor ideal (Sugiyono, 2012).

2.15 Penelitian Terkait

Berikut merupakan beberapa penelitian sebelumnya yang digunakan sebagai referensi penelitian dan pembanding dalam mengimplementasikan

algoritma *Fisher Yates Shuffle* dan algoritma *A Star* dalam pembuatan *game* yang dapat dilihat pada **Tabel 2.8** berikut ini.

Tabel 2.8 Kajian Penelitan Terkait

No.	Penulis	Judul	Keterangan
1	Siti Adha Zuliani dan Edy Winarno	Game Pembelajaran Membaca Iqra' Menggunakan Metode *Fisher Yates Shuffle*	Game edukasi pembelajaran iqra' merupakan *game* yang dibuat menggunakan metode *Fisher Yates Shuffle* dengan melakukan proses pengacakan soal pada *game.*
2	Ekojono, Dyah Ayu Irawati, Lugman Affandi dan Anugrah Nur Rahmanto	Penerapan Algoritma *Fisher-Yates* pada Pengacakan Soal *Game* Aritmatika	Penerapan algoritma *Fisher-Yates Shuffle* pada *game* aritmatika sebagai pengacak soal dan jawaban yang akan muncul dalam setiap permainan dan keluarnya soal tidak berulang.
3	Imam Ahmad dan Wahyu Widodo	Penerapan Algoritma *A Star* (A*) pada *Game* Petualangan Labirin Berbasis Android	Penerapan algoritma A* (*A Star*) untuk menu bantuan pada game petualangan labirin menggunakan *euclidean heuristic.*
4	Yuli Subarkah Wahyu Badriono	*Game* Labirin Pembelajaran Bahasa Jawa Menggunakan Algoritma *A Star* (A*) Berbasis Android	Pembuatan dan perancangan game Android serta penerapan algoritma *A Star* sehingga dapat digunakan sebagai media pembelajaran yang baik dan menyenangkan.
5	Bonita Deandra Risha Rukmana	Rancang Bangun *Game* Edukasi Sebagai Media	Pada penelitian ini metode *Fisher Yates Shuffle* digunakan sebagai pengacak

No.	Penulis	Judul	Keterangan
		Pembelajaran Adab-Adab Islami Menggunakan Algoritma *Fisher Yates Shuffle* Dan *A Star*	soal pada *game* sehingga soal dapat ditampilkan dengan urutan yang berbeda-beda dan algoritma *A Star* yang digunakan musuh untuk bergerak secara dinamis bergerak ke arah pemain.

Penelitian oleh Zuliani dan Winarno (2018) yang membuat sebuah *game* pembelajaran membaca iqra' menggunakan metode *Fisher Yates Shuffle* dengan melakukan proses pengacakan soal pada *game*. *Fisher Yates Shuffle* sangat penting dalam pembuatan *game* ini dan jika *Fisher Yates Shuffle* tidak diterapkan, maka *game* tersebut tidak berfungsi sebagaimana mestinya. Penelitian ini menghasilkan rancang bangun *game* edukasi belajar membaca iqro' serta mengembangkan *game* yang dapat memberikan hiburan sekaligus media pembelajaran tentang pengenalan huruf hijaiyah.

Ekojono (2017) melakukan penelitian tentang penerapan algoritma *Fisher-Yates* pada pengacakan soal *Game* Aritmatika. Kesimpulan yang didapatkan dari pembuatan *Game* Aritmatika ini adalah pengacakan menggunakan algoritma *Fisher Yates Shuffle* berhasil diterapkan di dalam *Game* Aritmatika sebagai pengacak soal dan jawaban yang akan muncul dalam setiap permainan dan keluarnya soal tidak berulang.

Ahmad dan Widodo (2017) melakukan penelitian tentang penerapan algoritma *A Star* (A*) pada *game* petualangan labirin berbasis Android. Penelitian ini menyajikan laporan hasil pengembangan *game* petualangan labirin yang menceritakan mengenai petualangan kelinci melewati sebuah labirin untuk mencari makanannya. AI (*Artificial Intelligence*) yang digunakan adalah algoritma A* dengan *euclidean distance* yang digunakan pada bantuan untuk melakukan pencarian jalur guna menemukan makanan kelinci. Hasil uji dari *game* petualangan labirin ini adalah jika pemain dalam kesusahan menemukan jalur menuju makanan kelinci, maka pemain dapat menggunakan tombol bantuan yang

akan dicarikan jalur terpendek oleh algoritma A* (*A Star*) dengan *euclidean distance* untuk menuju lokasi tempat makanan kelinci berada.

Badriono (2019) melakukan penelitian tentang pembuatan *game* labirin pembelajaran bahasa jawa menggunakan algoritma *A Star* (A*) berbasis Android. Permasalahan penelitian ini yaitu bagaimana membuat dan merancang *game* Android serta mengimplementasikan algoritma *A Star* sehingga dapat digunakan sebagai media pembelajaran yang baik dan menyenangkan. Hasil dari penelitian ini berupa *game* labirin pembelajaran bahasa Jawa berbasis Android yang memiliki fitur pembelajaran bahasa Jawa dari *game* labirin dan soal yang dapat membantu pembelajaran bahasa Jawa. Kesimpulan hasil penelitian ini adalah *game* pembelajaran bahasa Jawa dapat digunakan sebagai media pembelajaran yang baik dan menyenangkan sudah sesuai dan mudah digunakan untuk siswa dan siswi Sekolah Dasar.