BAB II

TINJAUAN PUSTAKA

2.1 Boxing

Menurut Shipley (2008), *boxing* merupakan bentuk gaya pertempuran tinju dengan menggunakan sarung yang diatur oleh peraturan international. Pada modern *boxing* terdapat peraturan yang tidak memperbolehkan menyerang dibawah pinggang dan tidak boleh menyerang juga saat lawan sudah terjatuh. *Fighting with the fists* merupakan bahasa Inggris kuno dari *boxing*. Peraturan pertama dipublikasikan oleh Jack Broughton (1743). Broughton mengajarkan untuk menggunakan "*mufflers*" pada *boxing*. Pertandingan *boxing* berlangsung di area yang dikelilingi oleh tali (the ring) dan dikelilingi oleh penonton. Kontestan menggunakan alat perlindungan, meliputi helm dan sarung tangan. Kontestan dikategorikan berdasarkan berat badan dari *flyweight* ke *heavyweight*.

2.2 Interaksi Manusia dan Komputer

Menurut Clegg (2008), *human-computer interaction* adalah pembelajaran tentang faktor-faktor manusia yang terasosiasi dengan desain teknologi, implementasi, dan penggunaaannya. Ketika *human-computer interaction* menjadi sub bidang penelitian *computer science* dan *psychology disciplines*, *human-computer interaction* juga muncul sebagai agenda pengembangan penelitian di dalam pembelajaran organisasi. Dalam kurun 20 tahun terakhir, revolusi teknologi telah mengubah tempat kerja dan memproduksi banyak konteks untuk memeriksa

fenomena *human-computer interaction*, meliputi desain tugas, pengiriman layanan, Internet, dan bermacam-macam platform komunikasi.

2.3 Game

Menurut Adams (2010), *game* adalah sebuah aktivitas bermain yang dilakukan dalam realitas yang tidak nyata. Peserta diminta untuk mencapai paling tidak satu tujuan *nontrivial* dengan bertindak sesuai dengan aturan yang ditetapkan. *game* berkembang dari keinginan manusia untuk bermain dan dari kemampuan manusia untuk berpura-pura. *Play* atau bermain merupakan sebuah kategori luas dari non esensial, umumnya untuk rekreasi. *Pretending* atau berpura-pura merupakan kemampuan mental untuk membangun realitas nasional yang tahu perbedaan dari dunia nyata dan yang berpura-pura bisa membuat, mengabaikan, atau mengubah sesuai keinginan. *Playing* dan *pretending* merupakan elemen *essential* dari bermain. Keduanya telah dipelajari secara ekstensif sebagai budaya dan fenomena psikologis.

Menurut Hogle(1996) ada beberapa keuntungan dengan belajar melalui *game* dengan paparan sebagai berikut.

1. Menstimulasi motivasi intrinsik dan meningkatkan minat, sifat alamiah rasa ingin tahu, ekspetasi, kontrol, interaksi, dan alur cerita fantasi pada *video games* dapat meningkatkan minat dan motivasi intrinsik pelajar, dan pelajar dapat mencoba untuk tetap berusaha menghadapi tantangan yang sulit untuk mendapatkan rasa pencapaian.
2. *Memory reserving*: dibandingkan dengan teknik pembelajaran *traditional*, belajar dengan *video games* dapat meningkatkan efek daya ingat lebih besar.

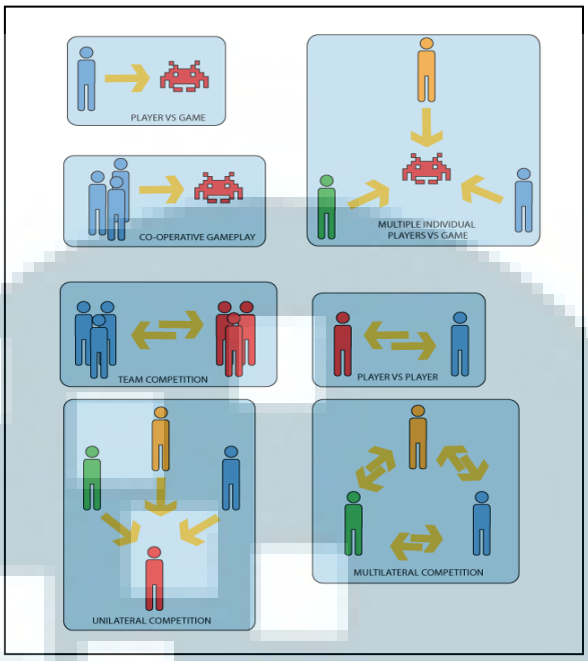
3. *Practice and feedbac*k: banyak *software* pembelajaran dengan *video games* memberikan pelajar kesempatan untuk mengulangi latihan dan secara cepat memberikan *feedback* jika *error*. Hal ini yang membuat pelajar menilai kemampuan belajar dan meningkatkan pencapaian objektif pembelajaran.
4. *Proving high-level thinking*: belajar menggunakan video *games* adalah cara belajar paling bagus karena desain video *games* mirip dengan struktur kognitif manusia yang membuat pelajar menemukan solusi-solusi dan mendapatkan pengetahuan memlalui penyelesaian masalah secara berulang, membuat keputusan, dan berintegrasi dengan apa yang dipelajari. Akhirnya, konten pembelajaran dapat secara konstan tertanam di ingatan pelajar.

2.4 Game Design

Menurut Nackel (2014), elemen-elemen formal adalah sesuatu yang membantu untuk membuat struktur sebuah *game*. Hubungan antar elemen-elemen formal inilah yang membangun sebuah *game*. Berikut adalah elemen-elemen tersebut.

1. Player

   Player merupakan sebuah *game design* yang meminta untuk *player* berinteraksi dengan *player* lainnya. *Player* bersifat sukarela dan aktif dalam kegiatan hiburan dan berpotensi untuk menjadi pemenang dalam kegiatan tersebut. Setiap *player* bisa memiliki peran yang berbeda saat bermain dan memiliki gaya bermain yang berbeda.



Gambar 2.1 *Player Interaction*
(Nacke, The Formal Systems of Games and Game Design Atoms, 2014)

2. Objectives

   Objectives adalah sesuatu yang memotivasi para *player* untuk ikut serta dalam *game*. *Objectives* dari sebuah *game* terlihat dapat dicapai tetapi menantang untuk mecapainya seperti mendapatkan *XP* (*experience point*) secara maksimal dengan bertahan hingga *level* pada *game* selesai. Para *player* harus menyelesaikan *objectives* sebagai ukuran keterlibatannya dalam *game* tersebut.

3. Procedures

   Procedures adalah aksi atau metode yang diperbolehkan untuk dilakukan oleh *rules* pada *game* dan bisa menjadi instruksi spesifik aksi yang harus dilakukan saat bermain. Pada *game* di komputer hal ini merupakan proses input dari *player*. Pada dasarnya harus menjawab *who does, what, where, when, and how*.

4. Rules

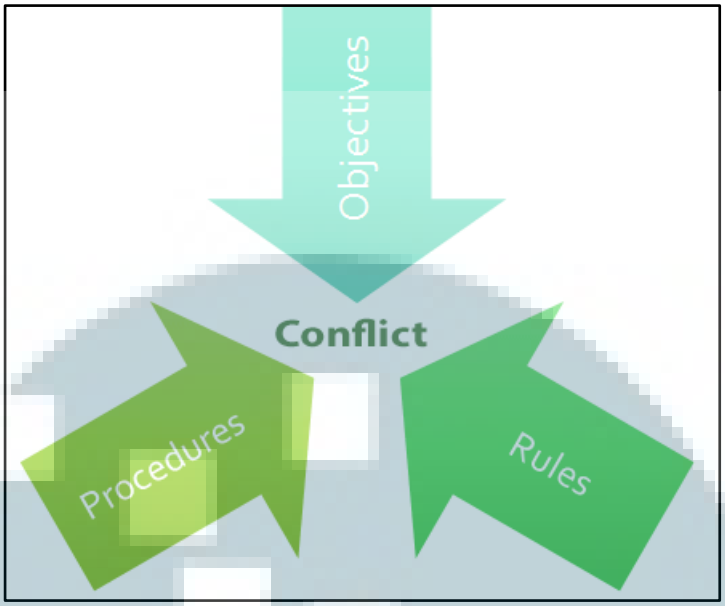
   Rules merupakan sebuah aturan yang terdapat pada dunia *game* yang mengatur apa saja yang dapat dan tidak dapat dilakukan oleh *player*. Tujuan utama dari pembuatan *rules* adalah mendefinisikan objek dan kondisi, membatasi aksi *player*, dan menentukan efek pada *player*. Jika *rules* tidak diikuti oleh *player*, *player* keluar dari *game*.

5. Resources

   Resources merupakan sebuah objek pada *game* yang memiliki nilai saat *player* berhasil mencapai suatu objektif individu. Nilai objek tersebut dapat ditentukan dari *scarcity* dan *utility*. *Utility* merupakan nilai yang dihitung dari seberapa berguna suatu objek dalam membantu *player* mencapai sebuah objektif. *Scarcity* merupakan nilai dari kelangkaan suatu objek.

6. Conflict

   Conflict adalah suatu keadaan yang muncul dari *procedures* dan *rules* yang menghalangi *player* untuk mencapai tujuannya. *Objectives* biasanya menuntun *player* menuju dalam *conflict* juga. Sebagai contoh c*onflict* dari *game first-person shooters* adalah bertahan hidup agar tidak terbunuh oleh *player* lain dan *conflict* dari Pinball adalah menjaga agar bola tidak keluar dari arena *game* hanya dengan menggunakan *flipper*.



Gambar 2.2 *Conflict*
(Nacke, The Formal Systems of Games and Game Design Atoms, 2014)

7. Boundaries

   Boundaries merupakan batas *game* dengan dunia nyata dan juga berhubungan dengan aksi yang hanya dapat dilakukan di *game* dan dapat mengakibatkan konsekuensi yang berbeda jika dilakukan di luar *game*.

8. Outcome

   Outcome adalah suatu hasil yang keluar saat *game* selesai. Hasil tersebut harus tidak pasti untuk menumbuhkan minat *player* dan biasanya dapat dihitung.

Menurut Gibson (2014), *dramatic element* adalah cerita dan narasi dari sebuah *game*, termasuk *premise*. *Dramatic element* membantu *player* untuk memahami *rules* dan mendorong *player* agar menjadi emosional terhadap *outcome* dari *game*.

Fullerton (2014) menyatakan bahwa *dramatic elements* memberikan *context* pada *gameplay*, melapisi dan mengintegrasikan *formal elements* dari sistem menjadi pengalaman yang bermakna. *Dramatic elements* yang dasar seperti *challenge* dan *play* selalu ditemukan di semua *game*. *Element* yang lebih rumit

seperti *premise*, *character*, dan *story* banyak digunakan di *game* lain untuk menjelaskan dan meningkatkan *elements* yang lebih abstrak dari sistem formal, menciptakan sensasi koneksi yang lebih dalam untuk *player* dan memperkaya pengalaman *player*.

1. Challenge

   Menurut Nackel (2014), *challenge* sangat individual dan ditentukan oleh kemampuan spesifik pemain. *Game designer* sering merujuk ke konsep yang dibuat oleh Csíkszentmihalyi(1990) yang disebut dengan *Flow*. *Flow* merupakan aktivitas mengimbangi antara *challenge*, *ability*, *frustration*, dan *boredom* untuk menghasilkan pengalaman sebuah pencapaian dan kesenangan.

2. Play

   Play merupakan *dramatic element* yang mendorong *player* untuk menjadi emosional dalam sebuah *game* dan juga merupakan kebebasan dalam bergerak di dalam struktur yang kaku.

3. Premise

   Premise merupakan elemen yang menetapkan aksi *game* dalam pengaturan atau metafora. Tanpa *premise* yang dramatis, *game* akan terlalu abstrak untuk *player* menjadi emosional di *outcome*-nya.

4. Character

   Character merupakan agen yang bertugas mengantarkan drama pada sebuah *game*. Dengan mengidentifikasi *character* dan *outcome* tujuan dari *character*, *audience* akan mengikuti *story's events*.

5. Story

Story merupakan *backstory* semacam versi rumit dari *premise*. *Backstory* memberikan pengaturan dan *context* untuk *conflict* pada *game* dan dapat menciptakan motivasi untuk karakter.

2.5 Unity

Menurut Ian (2012), Unity adalah sebuah *game engine* untuk membuat *game* 2D dan 3D dengan mengkombinasikan suara, gambar, animasi, dan lain-lainnya. Unity merupakan *game engine* yang dapat mempublikasikan *game* ke banyak *platform* (*multi-platform*) seperti Windows, Android, dan iOS. Unity menggunakan *MonoDevelop* untuk *editor scripting*. Menurut Patil dan Alvares (2015), *game engine* pada Unity berdasarkan pada bahasa Javascript dan C# dan *scripting* pada Unity dibangun berdasarkan Mono, implementasi *open-source* dari .NET Framework dan mendukung bahasa Boo, C, C++, C#, CIL, D, F#, Java, Oxygene, Python, Vala dan VB .NET. Skrip program pada Unity akan secara otomatis dikompilasi menjadi *file* DLL .NET dan kecepatan eksekusi javascript lebih cepat 20 kali daripada javascript traditional seperti yang dilansir oleh Essay, UK(2013). Menurut Patil dan Alvares (2015), *graphic engines* pada Unity memakai Direct3D (Windows), OpenGL (Mac, Windows, Linux), OpenGL ES(Android, iOS), dan proprietary APIs(Wii) dan juga ada tambahan pendukung *untuk bump mapping*, *reflection mapping*, *screen space ambient occlusion* (SSAO), *dynamic shadows* menggunakan *shadows maps*, *render-to-texture* dan efek-efek *full-screen post-processing*. Patil dan Alvares juga menyebutkan bahwa Unity mendukung aset-aset *art* dan *file* dengan format dari 3ds Max, Maya, Softimage, Blender, Modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop,

Adobe Fireworks dan Algorithmic Substance dan aset ini dapat dimasukkan ke dalam projek *game* melalui *graphical user Interface* pada Unity. Unity juga memiliki fitur-fitur *built-in* untuk membuat *game multiplayer* yang standar, yaitu *State Synchornization*, *Real-time Networking*, *Remote Procedure Calls*, dan *Backend Connectivity*. Di dalam Unity juga terdapat Unity Asset Server yang merupakan sebuah solusi *version control* untuk aset *game* dan skrip milik para *game developer*. Unity Asset Server menggunakan PostgreSQL sebagai *backend* dan sebuah sistem audio yang dibangun berdasarkan *library* FMOD yang memiliki kemampuan untuk *playback* audio terkompresi Ogg Vorbis dan Unity juga memiliki Unity Asset Store yang diluncurkan di bulan November 2010 yang memiliki paket aset lebih dari 7000, termasuk model 3D, *textures* dan *materials*, *particle systems*, *music* dan *sound effect*, *tutorial* dan *project*, paket *scripting*, *editor extension* dan *online services*.

2.6 Kinect

Menurut Tashev (2013), Kinect adalah sebuah alat interaksi *human-machine* yang menambahkan modalitas untuk palet dari desainer *player interface*: *gestures* dan *speech*. Kinect mengubah cara manusia berinteraksi dengan komputer, *kiosk*, dan alat *motion-controlled* lainnya dari aplikasi yang menyenangkan seperti bermain biola virtual, sampai ke aplikasi-aplikasi kesehatan dan terapi fisik, retail, edukasi, dan *training*. Kinect untuk Windows SDK dan *toolkit* memuat *driver*, *tools*, *APIs*, *device interfaces*, and *code samples* untuk memudahkan pengembangan aplikasi dengan Kinect. KDK sudah mengintegrasi *skeletal* dan *facial tracking* dan *gesture recognition*. *Voice recognition* menambahkan *dimension of human comprehension* dan *Kinect Fusion*

merekonstruksi data ke *3D* model. Gambar 2.3 merupakan Kinect versi 1 dan Gambar 2.4 menunjukkan Kinect versi 2.

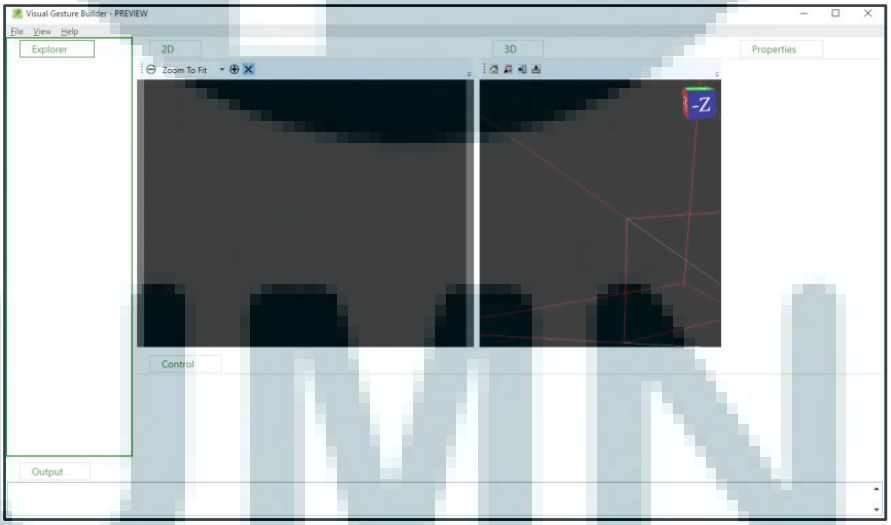
Gambar 2.3 Kinect Versi 1
(sumber: http://www.xbox.com/en-US/xbox-360/accessories/kinect)

Gambar 2.4 Kinect Versi 2
(sumber: http://www.xbox.com/en-US/xbox-one/accessories/kinect)

Microsoft telah menjelaskan bahwa Kinect versi 2 memiliki SDK yang lebih baik dari versi 1 dengan improvisasi orientasi pada *body*, *hand,* dan *joint* yang mampu melakukan tracking sebanyak 6 orang dengan 25 *skeletal joint* setiap orangnya termasuk *joint* yang baru seperti *hand tips*, *thumbs*, dan *shoulder center*. Improvisasi pemahaman *soft connective tissue* dan *body positioning* untuk hasil yang *crisp*, *avateering* yang lebih akurat, dan *avatar* terlihat lebih hidup. Gambar 2.5 merupakan Kinect Studio pada Kinect versi 2 yang telah diimprovisasi

recording dan *playback*-nya, dan juga pada Kinect versi 2 dapat membuat *custom gesture* dengan menggunakan Visual Gesture Builder seperti pada Gambar 2.6 yang akan di-*recognize* oleh sistem dan menggunakan *machine learning* untuk menambah produktivitas dan mengurangi *cost*.

Gambar 2.5 Kinect Studio
(sumber: https://developer.microsoft.com/en-us/windows/kinect/develop)



Gambar 2.6 Visual Gesture Builder yang Berhasil di-*Install*

2.7 Gesture Recognition

Menurut Quek (2004), penggunaan gerakan-isyarat, biasanya gerakan-isyarat tangan, yang bermaksud berkomunikasi dengan komputer dan mesin menarik untuk beberapa alasan. Pertama, banyak peneliti meneliti bahwa manusia memiliki fasilitas yang bagus untuk melakukan gerakan-isyarat dan selalu terjadi secara spontan. Kedua, tangan dan lengan selalu terhubung pada pertukaran interaksi manusia dan komputer, tidak perlu lagi untuk membuat *remote control* atau melengkapi manusa dengan alat komunikasi jika komputer bisa mengamati *player* dan bereaksi secara sesuai. Ketiga, ruang berinteraksi meluas dari satu layar ke banyak, dari layar kecil ke layar besar, dari 2D ke 3D, gerakan-isyarat menyajikan interaksi natural yang dapat mencocokkan penambahan ruang dan kematraan.

2.8 Algoritma Fisher-Yates Durstenfeld

Menurut Arndt (2009), Fisher-Yates Durstenfeld merupakan algoritma untuk membuat permutasi acak yang tidak bias oleh komputer dan menurut Wilson (2007) Fisher-Yates Durstenfeld juga disebut dengan "Knuth *shuffle*". Menurut Knuth (1997), algoritma shuffling ini dipublikasikan oleh R. A. Fisher dan F. Yates dalam bahasa biasa dan dibuat menjadi bahasa komputer oleh R. Durstenfeld. Cara kerja algoritma Fisher-Yates Durstenfeld adalah sebagai berikut.

1. Diberikan anggota kumpulan angka X_1, X_2, …, X_t yang akan di-*shuffle*.
2. Lakukan inisialisasi set $j \leftarrow t$.
3. *Generate* angka *random* U, yang terdistribusi antara 0 dan 1.

4. Pertukaran, set k ← ⌊jU⌋ + 1. Sekarang k adalah integer acak, antara 1 dan j.

   Tukar $X_k \leftrightarrow X_j$

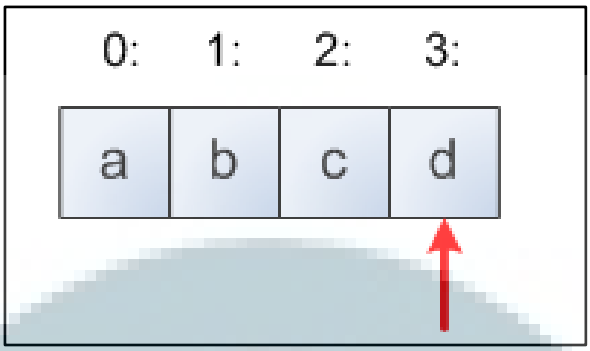
5. Kurangi j sebanyak 1. Jika j > 1, kembali ke step nomor 2.

Bendersky (2010) menjelaskan bagaimana cara kerja algoritma fisher-yates-durstenfeld dengan analogi topi pesulap. Topi pesulap yang di dalamnya bola yang berbeda-beda yaitu bola billiard lalu aduk bola-bola yang ada di dalam topi tersebut. Setelah itu, ambil bola tanpa melihat ke dalam topi dan jajarkan bola-bola tersebut dalam satu baris dengan asumsi adukan terjadi secara acak dan tidak dapat mengetahui angka bola hanya dengan menyentuh bola. Ketika topi sudah kosong, barisan bola-bola tersebut adalah permutasi acak bola. Tidak ada bola yang memiliki kemungkinan lebih besar untuk berada dalam baris pertama. Untuk mendapatkan baris kedua semua sisa bola mendapatkan kemungkinan yang sama juga. Gambar 2.7 merupakan *pseudocode* Fisher-Yates Durstenfeld dengan format Python, random 0 sampai dengan I karena *array* dimulai dari *index* 0.

```
def fisher_yates(arr):
    if len(arr) > 1:
        i = len(arr) - 1
        while i > 0:
            s = randint(0, i)
            arr[i], arr[s] = arr[s], arr[i]
            i -= 1
```

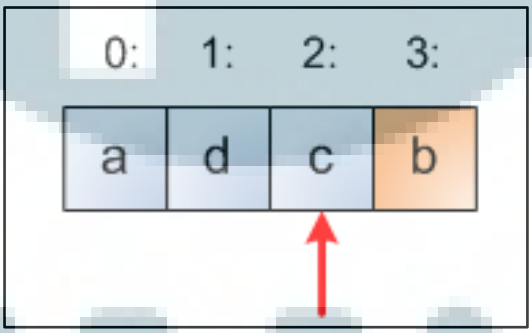
Gambar 2.7 Algoritma Fisher-Yates-Durstenfeld dengan Format Python
(Bendersky, The Intuition Behind Fisher-Yates Shuffling, 2010)

Trik yang dilakukan oleh Fisher-Yates adalah melakukan shuffling tanpa memori ekstra. Instruksi di bawah akan diperlihatkan secara bertahap. Dimulai dengan *array* yang berisi 4 elemen seperti pada Gambar 2.8.



Gambar 2.8 Isi *Array* Setelah Inisialisasi
(Bendersky, The Intuition Behind Fisher-Yates Shuffling, 2010)

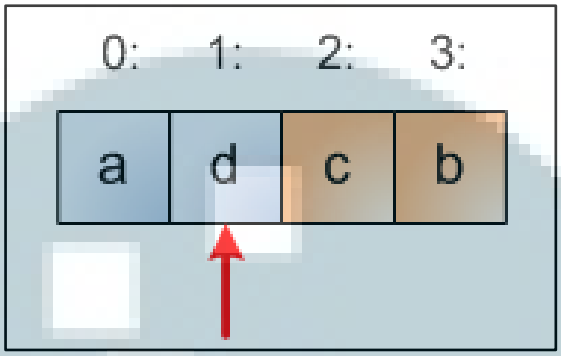
Terdapat huruf a, b, c, d dalam *array* dalam *index* [0:3]. Panah merah menunjukkan posisi *variable i* dalam perulangan. Langkah pertama di dalam perulangan adalah menentukan indeks secara *random* yang akan diambil dalam range [0:i]. Pada putaran awal [0:3] asumsi indeks 1 yang terpilih kemudian elemen indeks 1 ditukar dengan elemen 3. Gambar 2.9 menunjukkan isi *array* setelah putaran pertama perulangan selesai.



Gambar 2.9 Isi *Array* Setelah Putaran Pertama
(Bendersky, The Intuition Behind Fisher-Yates Shuffling, 2010)

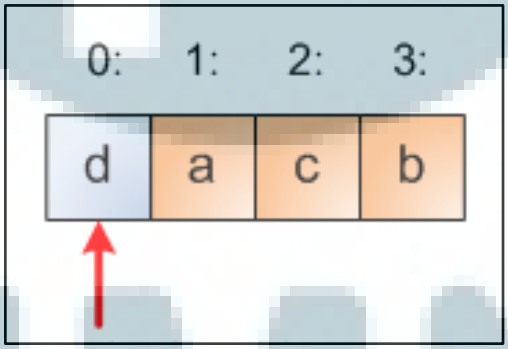
Kotak yang berwarna biru adalah topi pesulapnya dan kotak berwarna orange adalah kotak yang akan menampung permutasi acak. Pada putaran kedua *random range* indeks adalah [0:2]. Asumsi indeks 2 yang terpilih, biarkan indeks

2 tetap pada posisi originalnya karena sama dengan *i* yang isi *array*-nya dapat dilihat di Gambar 2.10.



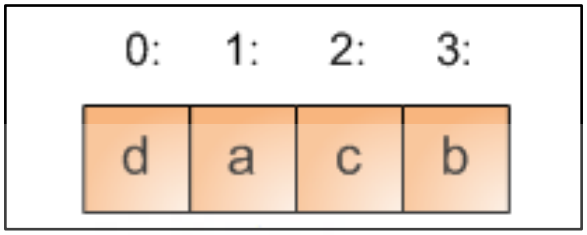
Gambar 2.10 Isi *Array* Setelah Putaran Kedua
(Bendersky, The Intuition Behind Fisher-Yates Shuffling, 2010)

Pada putaran ketiga, indeks 0 yang terpilih dari *random range* [0:1] kemudian lakukan pertukaran elemen 1 dengan elemen 0 seperti pada Gambar 2.11.



Gambar 2.11 Isi *Array* Setelah Putaran Ketiga
(Bendersky, The Intuition Behind Fisher-Yates Shuffling, 2010)

Ketika *conditional statement* perulangan $i > 0$, maka tinggal tersisa satu indeks lagi dan tidak perlu dilakukan *random* karena sudah pasti indeks tersebut terpilih dan algoritma selesai dengan hasil *array* seperti pada Gambar 2.12.

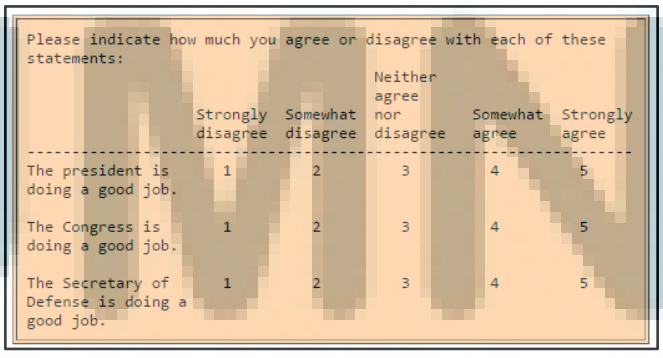


Gambar 2.12 Isi *Array* Setelah *Shuffle* Selesai
(Bendersky, The Intuition Behind Fisher-Yates Shuffling, 2010)

2.9 Likert Scale

Menurut Trochim (2006), *Likert Scaling* adalah *undimensional scaling*. Kelebihan dari *undimensional scaling* adalah secara umum mudah untuk dipahami. Contoh lebih banyak atau lebih sedikit, lebih tinggi atau lebih rendah, lebih berat atau lebih ringan.

Menurut Uebersax(2006), *Likert scale* adalah *multi-item scale* bukan *single item*. Likert scales awalnya dikembangkan oleh Rensis Likert, seorang sosiolog dari universitas Michigan dari tahun 1946 sampai dengan 1970. Likert ingin mengukur sikap psikologis dengan cara yang ilmiah. Secara khusus, Likert mencari sebuah metode yang akan menghasilkan langkah-langkah sikap yang cukup dapat diartikan sebagai pengukuran pada skala metrik yang tepat. Gambar 2.13 merupakan contoh *Likert scale*.



Please indicate how much you agree or disagree with each of these statements:

	Strongly disagree	Somewhat disagree	Neither agree nor disagree	Somewhat agree	Strongly agree
The president is doing a good job.	1	2	3	4	5
The Congress is doing a good job.	1	2	3	4	5
The Secretary of Defense is doing a good job.	1	2	3	4	5

Gambar 2.13 Contoh *Likert Scale*
(Handayani, Contoh Terapan Perhitungan Manual dan Analisa Hasil Kuesioner Menggunakan Skala Likert, 2014)

Handayani(2014) memberikan contoh terapan perhitungan manual dan analisa hasil kuesioner dengan *Likert Scale*. Dari hasil wawancara terstruktur yang dilakukannya terhadap 13 responden, maka diperoleh data sebagai berikut.

Alternatif Jawaban	Jumlah	Persentase
Sangat Setuju	2	15,4%
Setuju	4	30,80%
Tidak Setuju	7	53,80%
Sangat Tidak Setuju	0	0%
Total	**13 Orang**	**100%**

Gambar 2.14 Data Hasil Kuesioner Oleh Handayani (2014)
(Handayani, Contoh Terapan Perhitungan Manual dan Analisa Hasil Kuesioner Menggunakan Skala Likert, 2014)

Menurut Handayani(2014), perhitungan atas hasil wawancara menggunakan skala likert dapat dilakukan secara manual maupun menggunakan aplikasi SPSS dan LISREL. Berikut adalah contoh cara menghitung hasil pengamatan secara manual menggunakan penskoran *Likert Scale*.

1. Jumlah skor untuk 2 orang yang menjawab sangat setuju adalah 2 * 4 = 8.
2. Jumlah skor untuk 4 orang yang menjawab setuju adalah 4 * 3 = 12.
3. Jumlah skor untuk 7 orang yang menjawa tidak setuju adalah 7 * 2 = 14.
4. Jumlah skor untuk 0 orang yang menjawab sangat tidak setuju adalah 0 * 1= 0.
5. Total jumlah skor adalah 34.

Jumlah skor ideal untuk pertanyaan yang diajukan ke responden adalah sebagai berikut.

1. Skor tertinggi adalah 4 * 13 = 52 (sangat setuju).
2. Skor terendah adalah 1 * 13 = 13 (sangat tidak setuju)

Interpretasi skor hasil contoh pengamatan adalah 34 / 52 * 100% = 65,38%.