BAB 2
TINJAUAN PUSTAKA

2.1. Profil Perusahaan

Toko Zona Helm adalah salah satu toko yang menjual helm di daerah Kabupaten Bandung, khususnya didaerah Katapang. Toko ini dibuka sejak tahun 2014 dan telah menjual berbagai macam merek dan jenis helm, seperti helm *Half-Face, Full-Face, Retro, dan MotoCross*. Toko ini awalnya berada di kawasan Cimahi, namun karena sepinya pengunjung dan ramainya toko helm lain di daerah itu, membuat pemilik toko memilih untuk menutup toko di kawasan tersebut dan membuka kembali tokonya di kawasan Katapang. Toko Zona Helm yang kini berlokasi di Jl. Terusan Kopo No 196 KM 13,15 ini mulai berkembang dan penjualan helmnya mulai meningkat. Promosi yang dilakukan toko ini juga ikut meningkatkan pemasaran dan penjualan helm di toko Zona Helm ini.

2.2. Landasan Teori

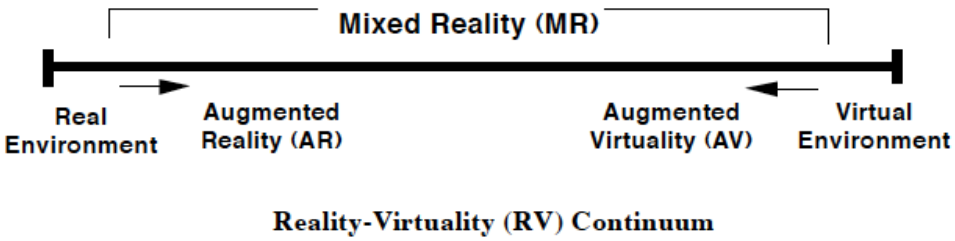
Landasan teori merupakan beberapa ulasan atau penjabaran mengenai teori-teori yang akan digunakan dalam membangun aplikasi ini, yang diambil dari berbagai sumber baik buku-buku, pustaka, maupun jurnal-jurnal ilmiah dan jurnal online. Landasan teori yang akan dibahas pada penelitian ini meliputi konsep *Augmented Reality, Face-Detection,* Citra Digital, Model Kamera dan Kalibrasi Kamera, DirectX, EmguCV, Pengujian Blackbox, UML (*Unified Modeling Language),* serta pembahasan singkat perangkat lunak dan *3D Model* pada aplikasi yang dibangun.

2.2.1. Augmented Reality

Menurut definisi Ronald Azuma (1997), ada tiga prinsip dari *augmented reality*. Yang pertama yaitu *augmented reality* merupakan penggabungan dunia nyata dan virtual, yang kedua berjalan secara interaktif dalam waktu nyata (*realtime*), dan yang ketiga terdapat integrasi antarbenda dalam tiga dimensi, yaitu benda maya terintegrasi dalam dunia nyata [3]. *Augmented reality* sendiri adalah teknologi yang menggabungkan benda maya dua dimensi dan ataupun tiga dimensi ke dalam sebuah lingkungan nyata lalu memproyeksikan benda-benda maya tersebut dalam waktu nyata. Benda-benda maya tersebut berfungsi untuk

menampilakan informasi yang tidak dapat diterima secara langsung oleh manusia. Hal ini membuat *Augmented Reality* digunakan sebagai alat untuk membantu persepsi dan interaksi pengguna dengan benda-benda maya tersebut.

Paul Milgram dan Fumio Kishino (1994) menjelaskan konsep *augmented reality* dalam teori mereka yang disebut dengan *Reality-Virtuality Continuum* yang dapat dilihat dalam Gambar 2.1 [4].



Gambar 2.1 *Reality-Virtuality Continuum*

Milgram dan Kishino merumuskan kerangka kemungkinan penggabungan dan peleburan dunia nyata dan dunia maya ke dalam sebuah kontinuum virtualitas. Sisi yang paling kiri adalah lingkungan nyata yang hanya berisi benda nyata, dan sisi paling kanan adalah lingkungan maya yang berisi benda maya. Dalam *augmented reality* atau realitas tertambah, yang lebih dekat ke sisi kiri, lingkungan bersifat nyata dan benda bersifat maya, sementara dalam *augmented virtuality* atau virtualitas tertambah, yang lebih dekat ke sisi kanan, lingkungan bersifat maya dan benda bersifat nyata. Realitas tertambah dan virtualitas tertambah digabungkan menjadi *mixed reality* atau realitas campuran.

Istilah *augmented reality* (AR) mulai dikenal pada tahun 1989, dimana awal pengembangannya berasal dari *Virtual Reality* yang diciptakan dan diperkenalkan oleh Jaron Lanier. Tahun 1992 *Augmented Reality* dikembangkan untuk dapat melakukan perbaikan pada pesawat *boeing*. Di tahun yang sama, LB Rosenberg mengembangkan sistem *Augmented Reality* yang digunakan di Angkatan Udara AS yang disebut *Virtual Fixtures*, dan pada tahun 1992 juga, Steven Feiner, Blair Maclntyre dan Dorée Seligmann, memperkenalkan untuk pertama kalinya *Major Paper* untuk perkembangan *Prototype Augmented Reality*.

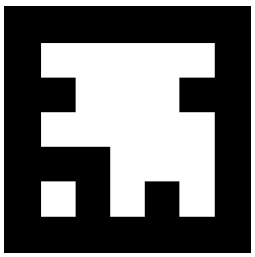
Pada tahun 1999, Hirokazu Kato, mengembangkan *ArToolkit* di HITLab dan didemonstrasikan di SIGGRAPH, pada tahun 2000, Bruce.H.Thomas, mengembangkan *ARQuake*, sebuah *Mobile Game Augmented Reality* yang ditunjukkan di *International Symposium on Wearable Computers.*

Pada tahun 2008, *Wikitude AR Travel Guide*, *memperkenalkan Android G1 Telephone* yang berteknologi *Augmented Reality*, tahun 2009, Saqoosha memperkenalkan *FLARToolkit* yang merupakan perkembangan dari *ArToolkit*. *FLARToolkit* memungkinkan kita memasang teknologi *Augmented Reality* disebuah website, karena *output* yang dihasilkan FLARToolkit berbentuk *Flash*. Ditahun yang sama, *Wikitude Drive* meluncurkan sistem navigasi berteknologi AR di Platform Android. Tahun 2010, *Acrossair* menggunakan teknologi *Augmented Reality* pada I-Phone 3GS [5].

Augmented Reality memiliki berbagai jenis teknik yang semakin berkembang sampai saat ini. Beberapa teknik-teknik tersebut adalah sebagai berikut :

1. Teknik *Marker Tracking (Marker Augmented Reality)*

   Teknik ini menggunakan sebuah *marker* untuk mendeteksi posisi tampilnya objek. *Marker* merupakan istilah yang dekat dengan dunia *augmented reality*. *Markers* adalah sebuah kotak hitam putih yang membuat sebuah aplikasi dapat melacak dan mendeteksi bentuk dan posisi dari objek 3D [6]. Marker yang dapat dideteksi oleh teknik marker tracking ini hanya yang berwarna hitam putih seperti Gambar 2.2.



Gambar 2.2 Contoh *Marker*

2. Teknik *Markerless Tracking (Markerless Augmented Reality)*

   Teknik ini masih menggunakan marker sebagai alat untuk mendeteksi. Namun marker yang digunakan dapat berbentuk apa saja tidak harus berwarnahitam putih, misalnya foto atau bahkan benda.

3. Teknik *GPS Tracking*

   Teknik GPS *Tracking* memanfaatkan lokasi *longitude* dan latitude sebagai alat untuk mendeteksi. Biasanya teknik ini digunakan di perangkat lunak untuk *smartphone*.

4. Teknik *Face Tracking*

   Teknik ini menggunakan wajah manusia untuk dideteksi dan pemanfaatannya bisa untuk *face tracking* maupun *face recognition.*

2.2.2. Deteksi Wajah (*Face Detection*)

Deteksi wajah merupakan salah satu teknik pengolahan citra, dimana inputnya merupakan citra wajah manusia dan menghasilkan outputnya berupa klasifikasi citra, dimana citra tersebut merupakan wajah manusia atau bukan. Deteksi wajah merupakan faktor utama dari beberapa bidang penelitian yang berkaitan dengan pemrosesan wajah. Beberapa bidang tersebut adalah sebagai berikut [7] :

1. Pengenalan wajah (*face recognition*) yaitu membandingkan citra wajah masukan dengan suatu database wajah dan menemukan wajah yang paling cocok dengan citra masukan tersebut.
2. Autentikasi wajah (*face authentication*) yaitu menguji keaslian/kesamaan suatu wajah dengan data wajah yang telah diinputkan sebelumnya.
3. Lokalisasi wajah (*face localization*) yaitu pendeteksian wajah namun dengan asumsi hanya ada satu wajah di dalam citra
4. Penjejakan wajah (*face tracking*) yaitu memperkirakan lokasi suatu wajah di dalam video secara real time.
5. Pengenalan ekspresi wajah (*facial expression recognition*) untuk mengenali kondisi emosi manusia.

Salah satu metode yang digunakan dalam mendekesi wajah adalah metode *Haar Cascade Classifier*. Metode ini merupakan pengembangan dari metode

deteksi objek *Viola-Jones*. Proses deteksi objek *Viola-Jones* adalah deteksi objek pertama yang menyediakan tingkat deteksi objek yang kompetitif secara *real-time* yang diusulkan pada tahun 2001 oleh Paul Viola dan Michael Jones [8]. Metode ini meskipun dapat dilatih untuk mendeteksi berbagai kelas objek, deteksi objek ini terutama didorong oleh masalah deteksi wajah. Metode *Haar Cascades Classifier* sendiri merupakan metode yang menggunakan statistical model (*classifier*). Pendekatan untuk mendeteksi wajah dalam gambar menggabungkan empat konsep utama :

1. Training data
2. Fitur segi empat sederhana yang disebut fitur Haar.
3. Integral image untuk pendeteksian fitur secara cepat.
4. Pengklasifikasi bertingkat (Cascade classifier) untuk menghubungkan banyak fitur secara efisien.

2.2.2.1. Training data pada *Haar*

Metode ini memerlukan 2 tipe gambar objek dalam proses training yang dilakukan, yaitu :

1. *Positive samples*

   Berisi gambar objek yang ingin dideteksi, apabila ingin mendeteksi mata maka positive samples ini berisi gambar mata, begitu juga objek lain yang ingin dikenali, misalnya hidung atau mulut.

2. *Negative samples*

   Berisi gambar objek selain objek yang ingin dikenali, umumnya berupa gambar background (tembok, pemandangan, lantai, dan gambar lainnya). Resolusi untuk sampel negatif disarankan untuk memiliki resolusi yang sama dengan resolusi kamera.

Training dari Haar menggunakan dua tipe sampel diatas. Informasi dari hasil training ini lalu dikonversi menjadi sebuah parameter model statistik.

2.2.2.2. Sistem Kerja Algoritma *Haar Cascade Classifier*

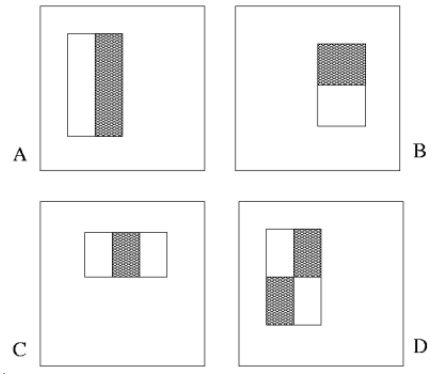
Algoritma Haar menggunakan metode statistikal dalam melakukan pengenalan wajah. Metode ini menggunakan *simple haar-like features* dan juga *cascade of boosted tree classifier*. Classifier ini menggunakan gambar berukuran

tetap. Cara kerja dari haar dalam mendeteksi wajah adalah menggunakan teknik *sliding window* pada keseluruhan gambar dan mencari apakah terdapat bagian dari gambar yang berbentuk seperti wajah atau tidak. *Haar* juga memiliki kemampuan untuk melakukan *scalling* sehingga dapat mendeteksi adanya mata yang berukuran lebih besar ataupun lebih kecil dari gambar pada classifier. Tiap fitur dari *haar like feature* didefinisikan pada bentuk dari fitur, diantaranya koordinat dari fitur dan juga ukuran dari fitur tersebut.

2.2.2.3. *Haar Feature*

Haar Feature adalah fitur yang digunakan oleh Viola dan Jones didasarkan pada *Wavelet Haar*. *Wavelet Haar* adalah gelombang tunggal bujur sangkar (satu interval tinggi dan satu interval rendah). Untuk dua dimensi, satu terang dan satu gelap. Selanjutnya kombinasi-kombinasi kotak yang digunakan untuk pendeteksian objek visual yang lebih baik. Ada tiga tipe kotak (*rectangular*) fitur [8] pada umunya dan dapat dilihat pada Gambar 2.3 yaitu:

1. Tipe two-rectangle feature (horizontal/vertikal)
2. Tipe three-rectangle feature
3. Tipe four-rectangle feature



Gambar 2.3 *Haar Feature*

Adanya fitur *Haar* ditentukan dengan cara mengurangi rata-rata piksel pada daerah gelap dari rata-rata piksel pada daerah terang. Jika nilai perbedaannya itu diatas nilai ambang atau *threshold*, maka dapat dikatakan bahwa fitur tersebut ada.

Nilai dari *haar-like feature* adalah perbedaan antara jumlah nilai-nilai piksel *gray level* dalam daerah kotak hitam dan daerah kotak putih. Kotak *haar-like feature* dapat dihitung secara cepat menggunakan *integral image*.

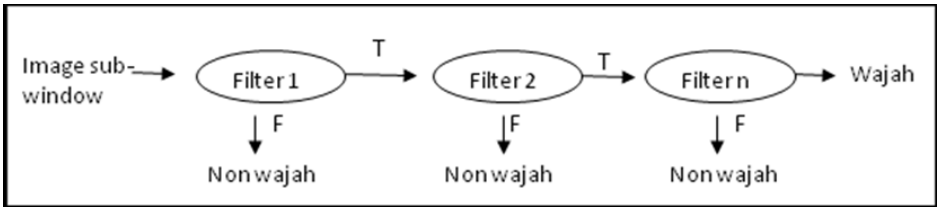
2.2.2.4. *Integral Image*

Integral Image digunakan untuk menentukan ada atau tidaknya dari ratusan fitur *Haar* pada sebuah gambar dan pada skala yang berbeda secara efisien. Pada umumnya, pengintegrasian tersebut berarti menambahkan unit-unit kecil secara bersamaan. Dalam hal ini unit-unit kecil tersebut adalah nilai-nilai piksel. Nilai integral untuk masing-masing piksel adalah jumlah dari semua piksel-piksel dari atas sampai bawah. Dimulai dari kiri atas sampai kanan bawah, keseluruhan gambar itu dapat dijumlahkan dengan beberapa operasi bilangan bulat per piksel.

2.2.2.5. *Cascade Classifier*

Cascade classifier adalah sebuah rantai *stage classifier*, dimana *setiap stage classifier* digunakan untuk mendeteksi apakah di dalam *image sub window* terdapat objek yang diinginkan (*object of interest*).

Stage classifier dibangun dengan menggunakan algoritma *adaptive-boost* (*AdaBoost*). Algoritma tersebut mengkombinasikan performance *banyak weak classifier* untuk menghasilkan *strong classifier*. *Weak classifier* dalam hal ini adalah nilai dari *haar-like feature*. Proses *Cascade Classifier* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Proses *Cascades Classifier*

2.2.3. Citra Digital

Citra digital merupakan suatu fungsi intensitas cahaya *f(x,y)*, dimana harga *x* dan *y* merupakan koordinat spasial dan harga fungsi tersebut pada setiap titik *(x,y)* merupakan tingkat kecemerlangan citra pada titik tersebut.

Citra digital yang tingginya N, lebarnya M, dan memiliki L derajak keabuan dapat diangkap sebagai fungsi :

$$f(x,y)\begin{cases}0 \le x \le M \\ 0 \le y \le N \\ 0 \le f \le L\end{cases} \qquad (1)$$

Citra ditital yang berukuran N x M lazim dinyatakan dengan matriks berukuran N baris dan M kolom sebagai berikut :

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & f(1,1) & \cdots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,M-1) \end{bmatrix} \quad (2)$$

Masing-masing element pada citra digital (berarti elemen matriks) disebut image elemtn atau piksel. Jadi, citra yang berukuran N x M mempunyai N.M buah piksel.

Proses digitalisai koordinat *(x,y)* dikenal sebagai pencuplikan citra (*image sampling*), sedangkat proses digitalisasi derajat keabuan *f(x,y)* disebut kuantisasi derajat keabuan (*gray-level quantization*).

Berdasarkan format penyimpanan nilai warnanya, citra terdiri atas empat jenis, yaitu :

1. Citra biner atau monokrom

Pada citra jenis ini, setiap titik atau piksel hanya bernilai 0 atau 1. Dimana setiap titik membutuhkan media penyimpanan sebesar 1 bit.

2. Citra skala keabuan

Citra skala keabuan mempunyai kemungkinan warna antara hitam (minimal) dan putih (makasimal). Jumlah maksimum warna sesuai dengan bit penyimpanan yang digunakan.

3. Citra warna

Setiap titik (piksel) pada citra warna mewakili warna yang merupkan kombinasi dari tiga warna dasar yaitu merah, hijai dan biru. Setiap warna dasar mempunyai intensitas sendiri dengan nilai maksimum 255 (8 bit). Setiap titik pada citra warna membutuhkan data 3 byte.

4. Citra warna berindeks

Setiap titik (piksel) pada citra warna berindeks mewakili indeks dari suatu tabel warna yang tersedia (biasanya disebut palet warna). Keuntungan pemakaian palet warna adalah kita dapat dengan cepat memanipulasi warna tanpa harus mengubah

informasi pada setiap titik dalam citra.Keuntungan lain penyimpanan lebih kecil. [9]

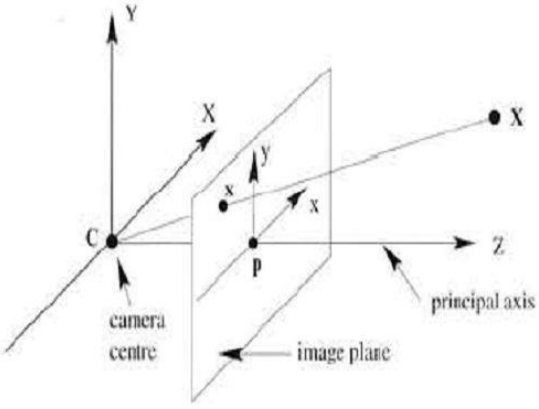
2.2.3.1. *Image Processing*

Picture Processing atau *image processing* merupakan suatu proses yang berkaitan dengan kegiatan memanipulasi dan menganalisis gambar atau citra yang dilakukan oleh komputer. Secara garis besar terdiri dari :

1. Digitalisasi dan Kompresi: Mengubah gambar menjadi bentuk diskrit; mengefisienkan pengkodean sehingga dapat menghemat ruang penyimpanan.
2. Pengingkatan, restorasi, dan rekontruksi : Meningkatkan gradasi (rendahnya contrast, blurr, noise) dari gambar; Memperbaiki gambar dari kerusakan.
3. Pencocokan, Pendeskripsian, dan rekognisi : Mencocokan gambar anara satu gambar dengan yang lain. Membagi gambar menjadi beberapa bagian, dan membandingkan hasil deskripsi terhadap model yang telah terdefinisi. [10]

2.2.4. Model Kamera dan Kalibrasi Kamera

Kamera dapat mengasilkan gambar 2D berasal dari gambar nyata (*word coordinat*) 3D. kamera mentransformasi media 3D (*word plane*) menjadi 2D (*image palane*) [11]. Model kamera salah satunya adalah *pinhole camera*, gambar yang dihasilkan adalah gambar perspektif yang mana kamera merupakan pusat proyeksi. Semua sinar yang datang akan dikumpulkan menjadi satu pada pusat kamera. Pada Model *pinhole camera* dapat dilihat pada Gambar 2.5.



Gambar 2.5 Model *Pinhole Camera*

Pada model *pinhole camera* sebuah titik yang terletak di dalam ruang nyata 3D diekspresikan sebagai titik $[x\ y\ z\ 1]^T$ di sistem koordinat dunia (*world coordinat*). Titik $[x\ y\ z\ 1]^T$ yang direkam dengan kamera akan mengalami transformasi perspektif dari ruang 3D sistem koordinat dunia keruang 2D sistem koordinat kamera (*image plane)*. Titik $[u\ v\ w]^T$ yang merupakan koordinat 2D pada koordinat kamera dapat diperoleh dengan persamaan koordinat homogeny sebagai berikut : [12]

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = K[I_3|O_3]\begin{bmatrix} R & -T \\ O_3^T & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = M\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (4)$$

Dimana M = [KR | - KRT]

Matriks K dan $[I_3 \mid O_3]$ pada persamaan (4) adalah parameter intrinsic yang mengandung variabel-variabel fisik pada kamera, seperti focus dan *principal point*. Matriks K adalah matriks kamera kalibrasi yang didefinisikan sebagai berikut :

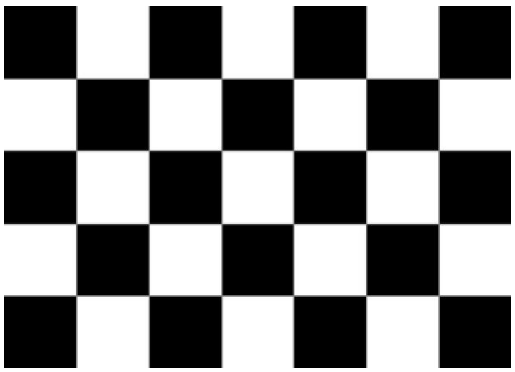
$$K = \begin{bmatrix} fa & 0 & u_0 \\ 0 & fb & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

Dimana *f* adalah panjang focus pada proyeksi kamera, *a* dan *b* adalah faktor konversi ukuran citra dari unit fisik (missal : cm atau mm) ke unit pixel pada sumbu *x* dan *y*, (u_0,v_0) adalah *principal point* pada proyeksi kamera.

Matriks $[I_3|O_3]$ didefinisikan sebagai : $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, matriks $\begin{bmatrix} R & -T \\ O_3^T & 1 \end{bmatrix}$ adalah parameter ektinsik kamera yang mendefinisikan lokasi dan orientasi kamera terhadap sistem koordinat dunia. Matriks translasi T berisi tiga buah elemen translasi dari pusat sistem koordinat dunia terhadap sistem koordinat kamera.

Proses kalibrasi kamera merupakan proses menentukan parameter-parameter pada kamera baik parameter intrinsik dan parameter ekstrinsik. Biasanya

proses kalibrasi kamera menggunakan media checkerboard yang diletakan pada sebuah papan. Langkah yang harus dilakukan dalam kalibrasi kamera adalah mengambil beberapa sampel dari *chessboard pattern* yang kemudan untuk setiap gambar tersebut dilakukan deteksi tepi pada *chessboard pattern* pada citra tersebut. Dengan mengetahui letak titik pada *world coodinat* dan *pixel coordinate* maka dapat ditemukan matriks yang mengubungkan keduanya. Dari proses kalibrasi kamera maka didapatkan matiks intrinsik dan intrinsik dari kamera. [13] *Chessboard Pattern* dapat dlihat pada Gambar 2.6.



Gambar 2.6 *Chessboard Pattern*

2.2.5. *DirectX*

DirectX adalah sekumpulan *Application Programming Interface* (API) yang dirancang untuk membantu pengembang aplikasi untuki menciptakan aplikasi game dan aplikasi multimedia berkinerja tinggi.

Fungsi utama *DirectX* adalah menyediakan mekanisme standar untuk mengakses berbagai perangkat keras yang berbeda dan mengisolasi aplikasi dari kerumitan implementasi ditingkat hardware.

Pada versi 9.0, *DirectX* terdiri atas komponen :

1. *DirectX Graphics (*Direct3D), menangani grafik 3D
2. *DirectX Audio, menangani efek suara.*
3. *DirectShow,* menangani multimedia playback.
4. *DirectInput, menangani input dari joystick, keyboard dan mouse.*
5. *DirectPlay,* menangani jaringan (untuk game multiplayer). [14]

2.2.5.1. Sejarah *DirectX*

Sebelum diluncurkan Windows 95, program game umumnya dirilis untuk sistem operasi *DOS* dengan bantuan *DOS* extender sepertdi *DOS4GW* untuk mengatasi keterbatasan memori yang disediakan *DOS.*

Keluarnya Windows 95 merupakan tanda dari awal berakhirnya era *DOS.* Banyak pengembang game mulai menulis ulang (memporting) game-game mereka agar bias berjalan optimal di Windows 95. *Graphic Device Interface* (GDI) Windows tidak didesain untuk game sehingga game-game yang menggunakan GDI untuk menangani grafis game tampil dengan grafis tidak bagus karena GDI sangat lambat.

Microsoft kemudan meluncurkan WinG (DIBSection) untuk mengatasi masalah pembambang game. WinG jauh lebih baik dibandingkan GDI karena lebih cepat, namun problem yang sering dihadapi pengembang game untuk memporting game mereka ke Windwos 95 adalah menampilkan grafis game secara full sreen. Dengan WinG aplikasi game ditampilkan dalam window seperti aplikasi-aplikasi lainnya, sedangkan game umurnya berjalan pada mode full screen.

Selain itu program game umumnya dibuat agar berjalan secepat mungkin, sehingga banyak fungsi-fungsi yang dimiliki dirancang untuk memanipulasi perangkat keras secara langsung. Untuk program game yang berjalan pada sistem operasi *DOS* hal ini tidak menjadi masalah karena *DOS* mengijinkan program mengakses perangkat keras secara langsung, namun untuk game yang berjalan pada sistem operasi multi-tasking seperti Windows mengakses perangkat keras secara langsung tidak disarankan mengingat tiap aplikasi harus berbagi sumber daya dengan aplikasi lain.

Untuk mengatasi hal ini *Microsoft* merilis *DirectX*. Awalnya *DirectX* dikembangkan oleh *RenderMorphics,* Microsoft kemudian ukut bergabung mengembangkan *DirectX*. Versi awal *DirectX* banyak dikritik oleh pengembang game karena desainnya yang buruk dan tidak terdokumentasi dengan baik.

Mulai versi 3 *Microsoft* lebih serius mengembangkan *DirectX* guna mendorong lebih banyak pengembang game untuk membuat game yang berjalan

pada sistem operasi Windows. *DirectX* pun akhirnya dirilis sebagai public domain sehingga *DirectX* dapat diperoleh dengan gratis.

Langkah *Microsoft* ini diikuti oleh produsen keras grafik dengan menyertakan dukungan terhadap *DirectX*. [14]

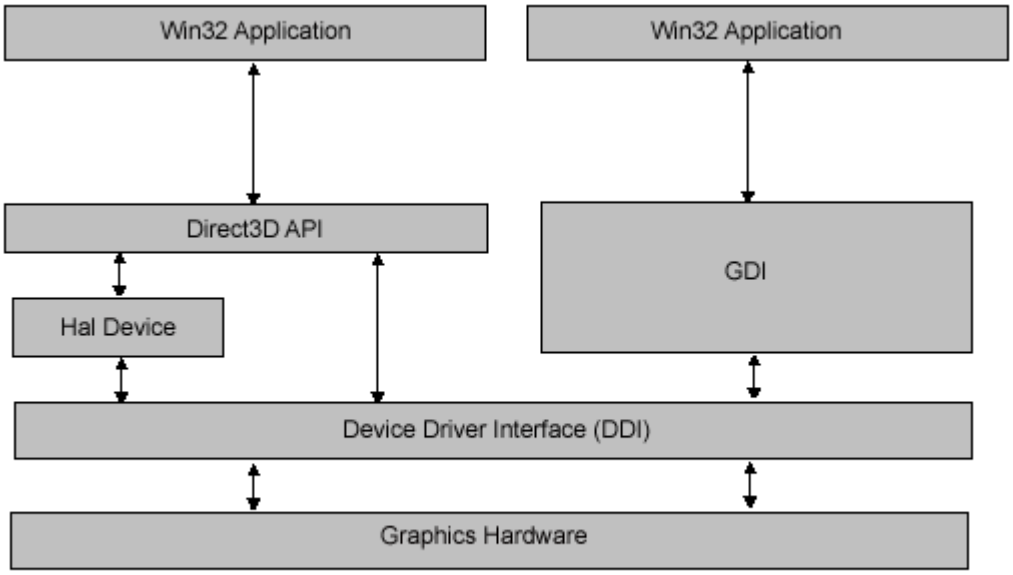
2.2.5.2. Arsitektur *Direct3D*

Direct3D adalah subsistem Windows yang sebanding dengan *Graphics Device Interface* (GDI). Berbeda dengan GDI, *Direct3D* hanya menyediakan layer tipis yang membungkus fungsional *graphic card*, tidak lebih. Anda tidak akan menemukan fungsionalitas untuk menggambar garis, menulis teks atau fungsional pemrograman grafis biasa anda temukan bila mengunakan *GDI*. Jika membutuhkan fungsionalitas semacam ini, anda harus membuat implementasinya

Dari Gambar 2.7, terlihat bahwa *Direct3D* berkomunikasi langsung dengan *hardware* melalui *HAL* (*Hardware Abstraction Layer).*

HAL atau *Hardware Abstraction Layer* adalah software layer tipis yang diimplementasikan oleh vendor hardware graphic card, berfungsi mengabstraksi device driver. *HAL* bersifat spesifik untuk tiap perangkat keras, tiap vendor yang ingin hardwarenya kompatibel dengan *Direct3D* harus membuat implementasi *HAL*.

HAL bertugas menerjemahkan perintah *Direct3D* agar dapat dieksekusi oleh device driver. Dengan cara ini maka *Direct3D* mampu berkomunikasi dengan beragam kartu grafis tanpa perlu tahu bagaimana cara kerja internal *hardware*. [14]

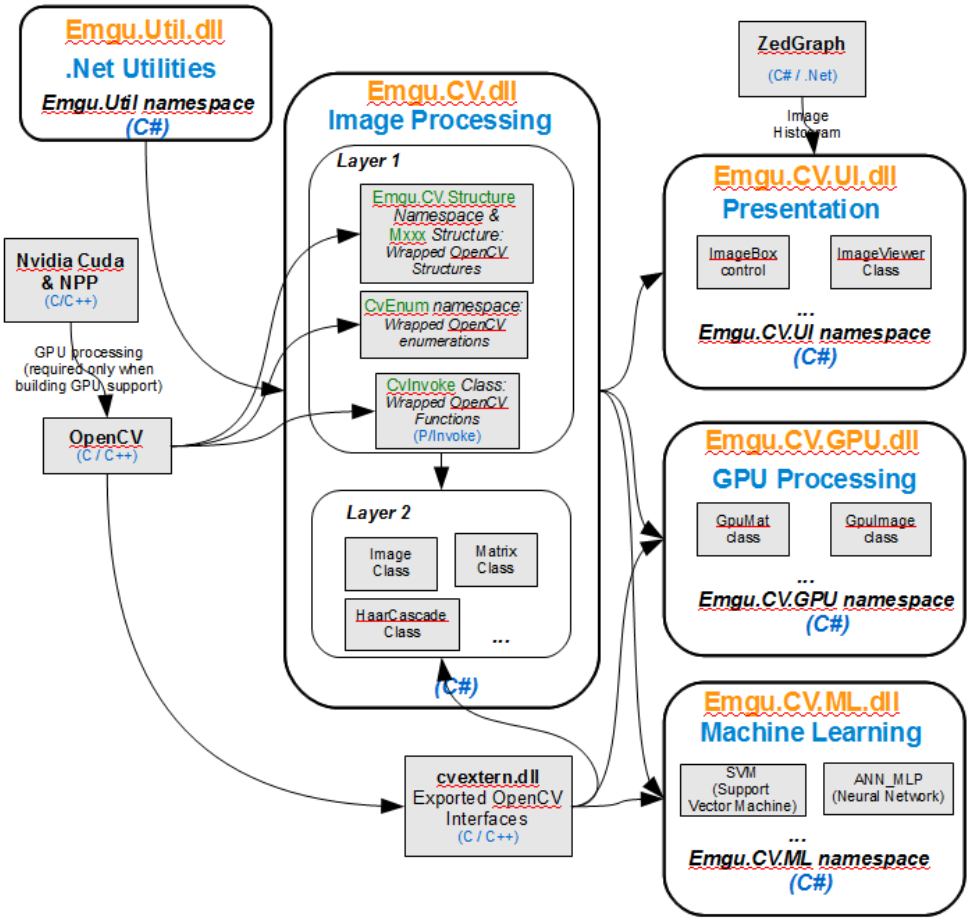


Gambar 2.7 Arsitektur *Direct3D*

2.2.6. *EmguCV*

EmguCV merupakan wrapper *cross platform* untuk *OpenCV* pada bahasa pemrograman *.NET*. *EmguCV* memungkinkan pemanggilan fungsi pada *OpenCV* pada bahasa pemrograman *.NET* seperti *c#, VB, VC++,IronPhython* dan lain-lain. *EmguCV* dapat di*compile* dengan Visual Studio, Xamarin Studio dan Unity, *EmguCV* dapat berjalan pada platform Windows, Linux, Mac OS X, iOS, Android dan Windows Phone.

EmguCV memiliki dua layer pada *wraper*nya seperi pada Gambar 2.8. Layer pertama mengandung pemetaan fungsi, struktur dan enumerasi yang menggambarkan itu pada *OpenCV*. Sedangkan layer kedua mengandung kelas-kelas yang dicampur dari *.NET*.



Gambar 2.8 Arsitektur *EmguCV*

2.2.7. Pengujian *Blackbox*

Pengujian yang dilakukan untuk antarmuka perangkat lunak, pengujian ini dilakukan untuk memperlihatkan bahwa fungsi-fungsi bekerja dengan baik dalam arti masukan yang diterima dengan benar dan keluaran yang dihasilkan benar-benar tepat, pengintegrasian dari eksternal data berjalan dengan baik.

Metode pengujian *blackbox* memfokuskan pada *requirement* fungsi dari perangkat lunak, pengujian ini merupkan komplenatari dari pengujian *whitebox.* Pengujian *whitebox dilakukan* terlebih dahulu pada proses pengujian, sedangkan pengujian *blackbox* dilakukan pada tahap akhir dari pengujian perangkat lunak. Proses yang terdapat dalam proses pengujian *blackbox* antara lain sebagai berikut:

1. Pembagian kelas data untuk pengujian setiap kasus yang muncul pada pengujian *whitebox.*
2. Analisis batasan nilai yang berlaku untuk setiap data.

2.2.8. *UML* (*Unified Modeling Language*)

UML dimulai secara resmi pada Oktober 1994, ketika Rumbaugh bergabung dengan Booch pada Relational Software Corporation. Proyek ini memfokuskan pada penyatuan metode Booch dan OMT. UML versi 0.8 merupakan metode penyatuan yang dirilis pada bulan Oktober 1995. Dalam waktu yang sama, Jacobson bergabung dengan Relational dan cakupan dari UML semakin luas sampai diluar perusahaan OOSE. Dokumentasi UML versi 0.9 akhirnya dirilis pada bulan Juni 1996.

UML disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat.

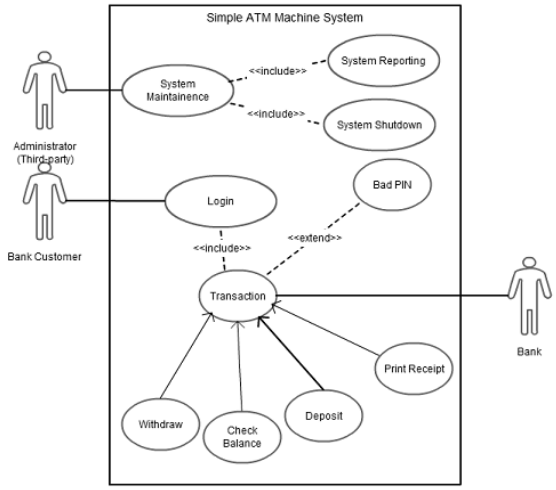
Bahasa pemodelan merupakan bagian terpenting dari metode. Ini merupakan bagian kunci tertentu untuk komunikasi. Jika anda ingin berdiskusi tentang desain dengan seseorang, maka Anda hanya membutuhkan bahasa pemodelan bukan proses yang digunakan untuk mendapatkan desain. UML merupakan bahasa standar untuk penulisan *blueprint software* yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian alat-alat dari sistem perangkat lunak. UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek. Beberapa diagram dalam UML adalah sebagai berikut [15]:

2.2.8.1. *Use Case Diagram*

Use case diagram merupakan permodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibangun. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibangun. *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Contoh *use case* diagram dapat dilihat pada Gambar 2.9. Ada dua hal utama pada *use case* yaitu aktor dan *use case.*

a. Aktor merupakan orang, proses atau sistem lain yang berinteraksi dengan sistem yang akan dibangun.
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau aktor.

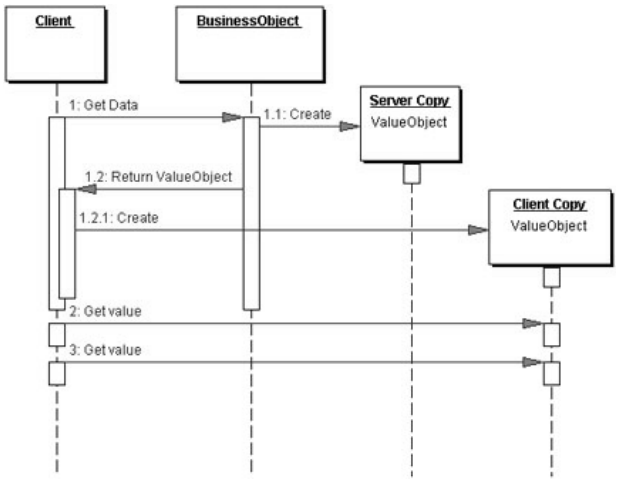


Gambar 2.9 Contoh *Use Case Diagram*

2.2.8.2. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa pesan yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah kejadian untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan simbol khusus untuk *objek boundary*, *controller* dan *persistent entity*. Contoh sequence diagram dapat dilihat pada Gambar 2.10.



Gambar 2.10 Contoh *Sequence Diagram*

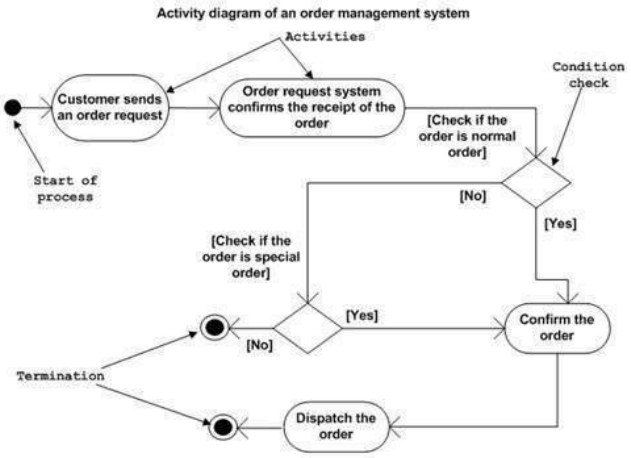
2.2.8.3. Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu, *activity diagram* tidak menggambarkan *behaviour internal* sebuah sistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Seperti halnya *state*, standar UML menggunakan segi empat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. Contoh dari *activity diagram* dapat dilihat pada Gambar 2.11.

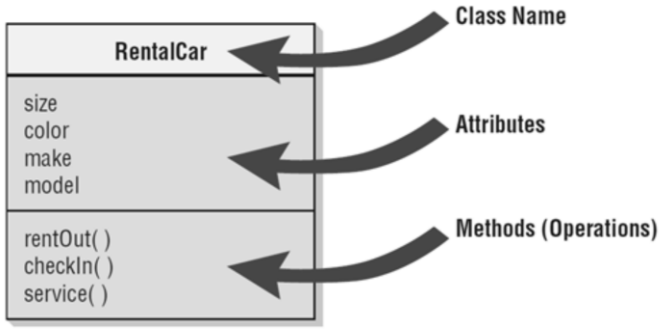


Gambar 2.11 Contoh *Activity Diagram*

2.2.8.4. *Class Diagram*

Class Diagram menggambarkan keadaan suatu sistem dengan menjelaskan keterhubungan antara suatu kelas dengan kelas yang lain yang terdapat pada sistem tersebut. *Class Diagram* bersifat statis. Di dalam *class diagram* digambarkan relasi dari masing - masing kelas tetapi tidak menggambarkan apa yang terjadi ketika kelas tersebut berelasi. Struktur dari *class diagram* dapat dilihat pada Gambar 2.12 adalah sebagai berikut :

- Nama Kelas/Stereotype
- Atribut, yaitu variabel-variabel yang dimiliki suatu kelas.
- Metoda, yaitu fungsi-fungsi yang dimiliki oleh suatu kelas.

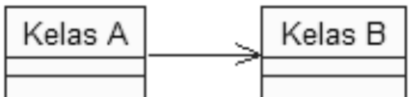


Gambar 2.12 *Class Diagram*

Class Diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class Diagram* memiliki 5 macam relasi yaitu *Association, Aggregation, Composition, Depedency,* dan *Generalization.*

a. *Association*

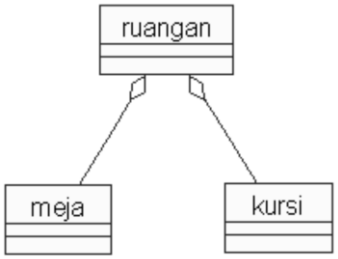
Assosiation adalah hubungan statis antar kelas. Umumnya menggambarkan kelas yang memiliki atribut berupa kelas lain, atau kelas yang harus mengetahui ekstensi kelas lain. Dalam notasi UML, asosiasi memiliki 2 jenis tipe yaitu asosiasi 2 arah (bidirectional) dan 1 arah (undirectional). Contoh dari asosiasi dapat dilihat pada Gambar 2.13.



Gambar 2.13 Relasi *Association*

b. *Aggregation*

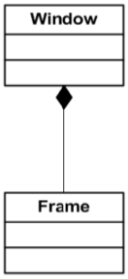
Aggregation adalah *association* dimana salah satu kelasnya merupakan bagian dari suatu kumpulan. *Aggregation* memiliki titik pusat yang mencakup keseluruhan bagian. Contoh dari *aggregation* dapat dilihat pada Gambar 2.14.



Gambar 2.14 Relasi *Aggregation*

c. *Composition*

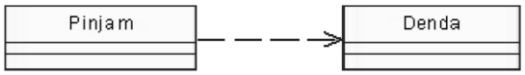
Composition ialah *aggregation* dengan ikatan yang lebih kuat. Di dalam *composite aggregation*, siklus hidup *part class* sangat bergantung pada *whole class* sehingga bila objek *instance* dari *whole class* dihapus maka objek *instance* dari *part class* juga akan terhapus. Contoh dari *composition* dapat dilihat pada Gambar 2.15.



Gambar 2.15 Relasi *Composition*

d. *Depedency*

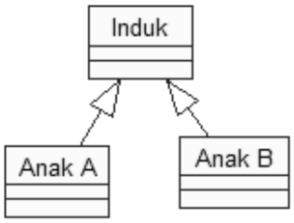
Depedency ialah hubungan antar-class di mana sebuah class memiliki ketergantungan pada class lainnya tetapi tidak sebaliknya. Contoh dari *depedency* dapat dilihat pada Gambar 2.16.



Gambar 2.16 Relasi *Depedency*

e. *Generalization*

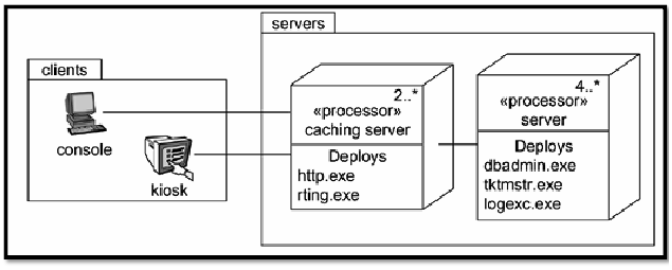
Generalization diperlukan untuk memperlihatkan hubungan pewarisan (*inheritance*) antar unsur dalam diagram kelas. Pewarisan memungkinkan suatu kelas mewarisi semua atribut, operasi ,relasi, dari kelas yang berada dalam hirarki pewarisannya. Contoh dari *generalization* dapat dilihat pada Gambar 2.17.



Gambar 2.17 Relasi *Generalization*

2.2.8.5. Deployment Diagram

Deployment diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak, bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifat fisik sebuah node adalah server, workstation, atau perangkat keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar node (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini. Contoh *deployment diagram* dapat dilihat pada Gambar 2.18.

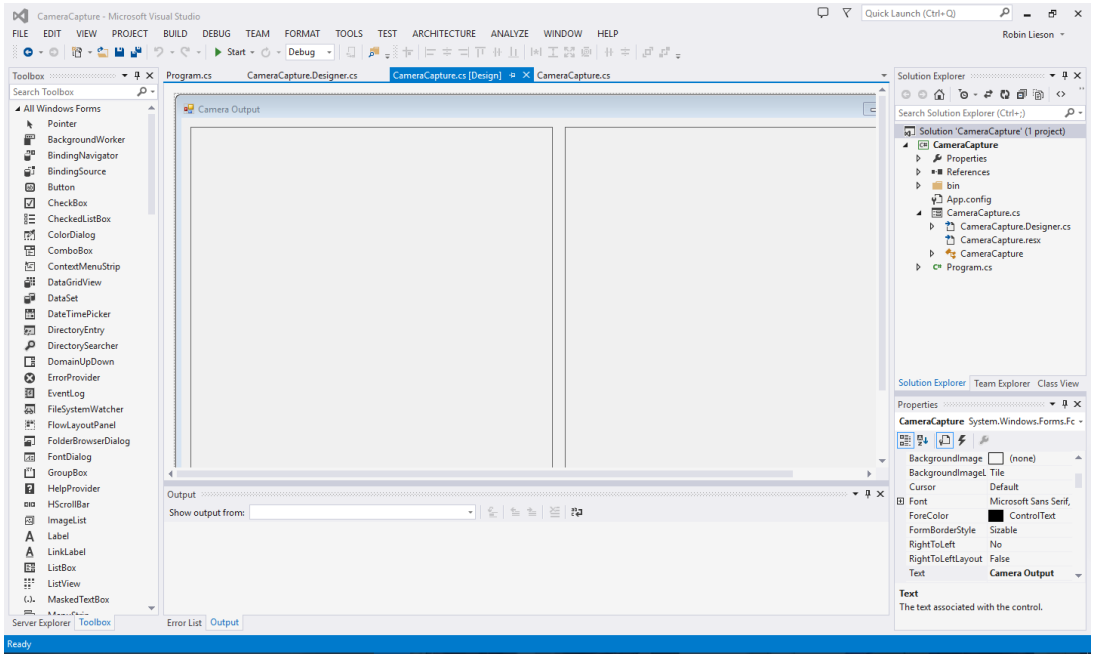


Gambar 2.18 Contoh *Deployment Diagram*

2.2.9. Microsoft Visual Studio

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi. Baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya dalam bentuk aplikasi *console*, aplikasi *windows*, ataupun aplikasi *web*. Visual Studio mencakup *compiler*, SDK, *Integrated Development Environment* (IDE), dan dokumentasi (umumnya berupa *MSDN Library*). *Compiler* yang berada dalam paket Visual Studio antara lain *Visual C++*, *Visual C#*, *Visual Basic*, *Visual Basic .NET*, *Visual InterDev*, *Visual J++*, *Visual J#*, *Visual FoxPro*, dan *Visual SourceSafe*.

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di *Windows*) ataupun *managed code* (dalam bentuk *Microsoft Intermediate Language* di atas .NET *Framework*). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *Windows Mobile* (yang berjalan di atas *.NET Compact Framework*) [16]. Tampilan antarmuka Visual Studio dapat dilihat pada Gambar 2.19.



Gambar 2.19 Antarmuka Microsoft Visual Studio

2.2.10. Blender 3D

Blender adalah perangkat lunak untuk desaign grafis yang digunakan untuk membuat film animasi, efek visual, aplikasi 3D interaktif atau video game. Fitur Blender termasuk pemodelan 3D, unwrapping UV, texturing, rigging dan menguliti, cairan dan simulasi asap, simulasi partikel, simulasi tubuh lembut, menjiwai, cocok bergerak, pelacakan kamera, rendering, video editing dan compositing. Ia juga memiliki mesin permainan (*game engine*) *built-in*. Blender menggunakan *OpenGL* sebagai render grafiknya yang dapat digunakan pada berbagai macam sistem operasi seperti *Windows*, *Linux* dan *Mac OS X*.

Blender dikembangkan sebagai aplikasi *in-house* oleh studio animasi Belanda Neo Geo pada tahun 1988, dimana Ton Roosendaal bertangung jawab sebagai *art director* dan pengembang *software internal.* Pada tahun 1998, Ton mendirikan perusahaan yang bernama Not a Number (NaN) untuk mengembangkan dan memasarkan Blender lebih jauh. Cita–cita NaN adalah untuk menciptakan sebuah software animasi 3D yang padat, *cross platform* yang gratis dan dapat digunakan oleh masyarakat komputer secara umum. Blender awalnya didistribusikan sebagai *shareware* hingga akhirnya NaN bangkrut pada tahun 2002.

Tahun 2001 NaN dibentuk ulang menjadi perusahaan yang lebih kecil. NaN kemudian meluncurkan software komersial pertamanya, Blender Publisher. Sasaran pasar software ini adalah untuk web 3D interaktif. Angka penjualan yang rendah dan iklim ekonomi yang tidak menguntungkan saat itu mengakibatkan NaN ditutup. Penutupan ini termasuk penghentian terhadap pengembangan Blender. Ton Roosendaal akhirnya mendirikan organisasi non profit yang bernama Blender Foundation agar Blender tidak hilang dari pasar *software* animasi 3D. Tujuan utama Blender Foundation adalah terus mempromosikan dan mengembangkan Blender sebagai proyek *open source*. Pada tahun 2002 Blender dirilis ulang dibawah syarat–syarat *GNU General Public License* [17]. Tampilan antarmuka Blender dapat dilihat pada Gambar 2.20.



Gambar 2.20 Antarmuka Blender 3D