DAFTAR PUSTAKA

Yunita, 2012. “Pengaruh Metode Bermain Peran (Role Playing) Terhadap Minat Belajar Sejarah Siswa Kelas VIII Di Mts Swasta Bastanul Falah Kec Sei Dadap”. Medan: Universitas Negeri Medan

Umamah, N. 2014. Kurikulum 2013 dan Kendala yang Dihadapi Pendidik dalam Merancang Desain Pembelajaran Sejarah. Prosiding: Seminar Nasional Pembelajaran Sejarah ditengah Perubahan. Malang: Universitas Negeri Malang.

Isjoni. 2007. Pembelajaran Sejarah Pada Satuan Pendidikan. Bandung: Alfabeta.

Kochar, K. S. 2008. Pembelajaran Sejarah. Jakarta: Gramedia Widiasarana Indonesia.

Sapriya. 2009. Pendidikan IPS. Bandung: PT Remaja Rosda Karya.

Na'im, M., dan Sumardi. 2017. The Development of Digital Module Through eXe Application Based to Improve Learners Attraction and Learning Outcomes of Indonesia History. The International Journal of Social Sciences and Humanities Invention.

Umamah, N. 2015. Teachers, Innovative, Instructional Design and a Good Character in Information Era. Proceeding of Internasional Seminar Education for Nation Character Building. Tulungagung: STKIP PGRI Tulungagung

Rahmadani, Isa. 2017. “Rancang Bangun Aplikasi Game Edukasi Bahasa Bugis Dan Bahasa Makassar Berbasis Android”. UIM Alauddin Makassar. Indonesia.

Rasyad, M. Fuad Afif. 2017. “Perancangan Video Game sebagai Media Edukasi dalam Pembelajaran Sistem periodic Unsur pada Mata pelajaran Kimia”. Universitas Hasanuddin Makassar, Indonesia.

Radion Kristo Purba. Rini Nur Hasanah, M. Azis Muslim. Jurnal EECCIS Vol. 7. 2013. “Implementasi Logika Fuzzy Untuk Mengatur Perilaku Musuh dalam Game Bertipe Action-RPG.

Wulandari, A.D. 2012. Game Edukatif Sejarah Komputer Menggunakan Role Playing Game (RPG) Mkaer XP Sebagai Media Pembelajaran di SMP Negeri 2 Kalibawang. Universitas Negeri Yogyakarta.

Nurmansyah, Niman. 2012. Pembangunan Game V-Moe Attack. Universitas Komputer.

Tridonanto. Al. “Optimalkan Potensi Anak Dengan *Game”.* Jakarta: Elex Media Komputindo, 2011

Echols, John. M. 1996. Kamus Inggris-Indonesia. Jakarta: Gramedia

Ismail, A. 2006. Education Games. Yogyakarta: Pilar Media.

Heriyanto Edi, Erma Kumalasari Nurnawati, Dina Andayati. 2018. “Skripsi Implementasi Kecerdasan Buatan Pada Game Menggunakan Metode Pathfinding Dengan Game Engine Unity3D”. Yogyakarta.

Niagahoster: https://www.niagahoster.co.id/blog/plugin-adalah/. Diakses pada tanggal 28 maret 2020.

Fadhil Aan, Hanif Al Fatta. 2015. “Kecerdasan Buatan Pada Bot Dalam Game Escape Alcatraz”. Yogyakarta: AMIKOM.

Alfianti Chaulina Oktavia, Rakhmad Maulidi. 2019. “Penerapan Logika Fuzzy Sugeno Untuk Penentuan Reward Pada Game Edukasi Aku Bisa”. Malang: STIKI Malang.

Kusumaningrum Triana. 2016. “Pengembangan Game Edukasi Berbasis Android Untuk Belajar Kosakata Bahasa Prancis Di Sma Negeri 2 Klaten”. Indonesia: Universitas Negeri Yogyakarta.

Anton Jarot Haryasena. 2013. “Perancangan Game Edukasi Untuk Anak Usia Dini (4-6 Tahun) Berbasis Android. Indonesia: Universitas Muhammadiyah Surakarta.

Sardiman AM. 2012. “Pembelajaran Sejarah Dan Nilai-Nilai Kepahlawanan”.

Oseady Rian Prahastito. 2016. “Aplikasi Game Edukasi Budaya Dan Aksara Lampung Berbasis Android”. Indonesia: Universitas Lampung.

Andarista Pratika. 2018. “Implementasi Non Player Character Pada Game “The Lost Baby” Menggunakan Metode Finite State Machine”. Institute Teknologi Nasional Malang.

https://www.suara.com/news/2020/12/17/152750/biografi-sultan-hasanuddin-si-ayam-jantan-dari-timur?page=all. Diakses pada tanggal 24 april 2021

https://glints.com/id/lowongan/pengertian-dan-prinsip-game-design/#.YLSAnfkzbDc. Diakses pada tanggal 25 april 2021

Hariadi Fajar. 2015. “Perhitungan Damage Dan Experience Dinamis Berdasarkan Kemampuan Pemain Menggunakan Fuzzy Inference System”. Institut Teknologi Sepuluh Nopember Surabaya

LAMPIRAN

LAMPIRAN 1

Source code battle AICore

```
var Imported = Imported || {};
Imported.YEP_BattleAICore = true;

var Yanfly = Yanfly || {};
Yanfly.CoreAI = Yanfly.CoreAI || {};
Yanfly.CoreAI.version = 1.15;
//=============================================================================
// Parameter Variables
//=============================================================================
Yanfly.Parameters = PluginManager.parameters('YEP_BattleAICore');
Yanfly.Param = Yanfly.Param || {};

Yanfly.Param.CoreAIDynamic = String(Yanfly.Parameters['Dynamic Actions']);
Yanfly.Param.CoreAIDynamic = eval(Yanfly.Param.CoreAIDynamic);
Yanfly.Param.CoreAIDynTurnCnt = String(Yanfly.Parameters['Dynamic Turn Count']);
Yanfly.Param.CoreAIDynTurnCnt = eval(Yanfly.Param.CoreAIDynTurnCnt);
Yanfly.Param.CoreAIElementTest = String(Yanfly.Parameters['Element Testing']);
Yanfly.Param.CoreAIElementTest = eval(Yanfly.Param.CoreAIElementTest);
Yanfly.Param.CoreAIDefaultLevel = Number(Yanfly.Parameters['Default AI Level']);

//=============================================================================
// DataManager
//=============================================================================

Yanfly.CoreAI.DataManager_isDatabaseLoaded = DataManager.isDatabaseLoaded;
DataManager.isDatabaseLoaded = function() {
  if (!Yanfly.CoreAI.DataManager_isDatabaseLoaded.call(this)) return false;
  if (!Yanfly._loaded_YEP_BattleAICore) {
    this.processCoreAINotetags1($dataEnemies);
```

```
        this.processCoreAINotetags2($dataSkills);
    this.processCoreAINotetags3($dataStates);
    this.processCoreAINotetags4($dataSystem);
    Yanfly._loaded_YEP_BattleAICore = true;
  }
        return true;
};
```

```
DataManager.processCoreAINotetags1 = function(group) {
  var note1 = /<(?:AI PRIORITY)>/i;
  var note2 = /<\/(?:AI PRIORITY)>/i;
  var note3 = /<(?:AI CONSIDER TAUNT|ai considers taunts)>/i;
  var note4 = /<(?:AI LEVEL):[ ](\d+)>/i;
  for (var n = 1; n < group.length; n++) {
                var obj = group[n];
                var notedata = obj.note.split(/[\r\n]+/);

    obj.aiPattern = [];
    var aiPatternFlag = false;
    obj.aiConsiderTaunt = false;
    obj.aiLevel = Yanfly.Param.CoreAIDefaultLevel * 0.01;

                for (var i = 0; i < notedata.length; i++) {
                        var line = notedata[i];
                        if (line.match(note1)) {
                                aiPatternFlag = true;
                        } else if (line.match(note2)) {
        aiPatternFlag = false;
      } else if (aiPatternFlag) {
        obj.aiPattern.push(line);
      } else if (line.match(note3)) {
        obj.aiConsiderTaunt = true;
      } else if (line.match(note4)) {
        obj.aiLevel = parseFloat(RegExp.$1 * 0.01);
```

```
      }
                }
        }
};

DataManager.processCoreAINotetags2 = function(group) {
        if (Yanfly.SkillIdRef) return;
  Yanfly.SkillIdRef = {};
  for (var n = 1; n < group.length; n++) {
                var obj = group[n];
    if (obj.name.length <= 0) continue;
                Yanfly.SkillIdRef[obj.name.toUpperCase()] = n;
        }
};

DataManager.processCoreAINotetags3 = function(group) {
        if (Yanfly.StateIdRef) return;
  Yanfly.StateIdRef = {};
  for (var n = 1; n < group.length; n++) {
                var obj = group[n];
    if (obj.name.length <= 0) continue;
                Yanfly.StateIdRef[obj.name.toUpperCase()] = n;
        }
};

DataManager.processCoreAINotetags4 = function(group) {
        if (Yanfly.ElementIdRef) return;
  Yanfly.ElementIdRef = {};
  for (var i = 0; i < group.elements.length; ++i) {
    var obj = group.elements[i].toUpperCase();
    if (typeof obj === 'string' && obj !== '') Yanfly.ElementIdRef[obj] = i;
  }
};
```

```
//=============================================================================
// BattleManager
//=============================================================================

if (Yanfly.Param.CoreAIDynamic) {
  Yanfly.CoreAI.BattleManager_getNextSubject =
      BattleManager.getNextSubject;
  BattleManager.getNextSubject = function() {
    //this.updateAIPatterns();
    var battler = Yanfly.CoreAI.BattleManager_getNextSubject.call(this);
    if (battler && battler.isEnemy()) battler.setAIPattern();
    return battler;
  };
};

BattleManager.updateAIPatterns = function() {
    $gameTroop.updateAIPatterns()
};

Yanfly.CoreAI.BattleManager_isInputting = BattleManager.isInputting;
BattleManager.isInputting = function() {
  if ($gameTemp._tauntMode) return false;
  return Yanfly.CoreAI.BattleManager_isInputting.call(this);
};

//=============================================================================
// Game_Battler
//=============================================================================

Object.defineProperty(Game_Battler.prototype, 'level', {
    get: function() {
        return this.getLevel();
    },
    configurable: true
```

```
});

if (!Game_Battler.prototype.getLevel) {
  Game_Battler.prototype.getLevel = function() {
      return $gameTroop.highestLevel();
  };
};

Game_Battler.prototype.setAIPattern = function() {
    Game_Battler.prototype.makeActions.call(this);
};

Game_Battler.prototype.aiConsiderTaunt = function() {
  return false;
};

Game_Battler.prototype.hasSkill = function(skillId) {
    return this.skills().contains($dataSkills[skillId]);
};

Game_Battler.prototype.hasState = function(stateId) {
    return this.states().contains($dataStates[stateId]);
};

Game_Battler.prototype.notState = function(stateId) {
    return !this.isStateAffected(stateId);
};

Game_Battler.prototype.aiLevel = function() {
    return Yanfly.Param.CoreAIDefaultLevel * 0.01;
};

//=============================================================================
// Game_Enemy
```

```
//=============================================================================


Game_Enemy.prototype.skills = function() {
  var skills = []
  for (var i = 0; i < this.enemy().actions.length; ++i) {
    var skill = $dataSkills[this.enemy().actions[i].skillId]
    if (skill) skills.push(skill);
  }
  skills = AIManager.getPatternSkills(skills, this.enemy().aiPattern);
  return skills;
};


Yanfly.CoreAI.Game_Enemy_makeActions = Game_Enemy.prototype.makeActions;
Game_Enemy.prototype.makeActions = function() {
    if (this.enemy().aiPattern.length > 0) {
      this.setAIPattern();
      this.setActionState('waiting');
    } else {
      Yanfly.CoreAI.Game_Enemy_makeActions.call(this);
    }
};


Game_Enemy.prototype.aiConsiderTaunt = function() {
  if (!Imported.YEP_Taunt) return false;
  return this.enemy().aiConsiderTaunt;
};


Game_Enemy.prototype.setAIPattern = function() {
    Game_Battler.prototype.setAIPattern.call(this);
    if (this.numActions() <= 0) return;
    AIManager.setBattler(this);
    for (var i = 0; i < this.enemy().aiPattern.length; ++i) {
      if (Math.random() > this.aiLevel()) continue;
      var line = this.enemy().aiPattern[i];
```

```
      if (AIManager.isDecidedActionAI(line)) return;
    }
    Yanfly.CoreAI.Game_Enemy_makeActions.call(this);
};

Game_Enemy.prototype.aiLevel = function() {
    return this.enemy().aiLevel;
};

//=============================================================================
// Game_Unit
//=============================================================================

Game_Unit.prototype.highestLevel = function() {
    return $gameParty.highestLevel();
};

Game_Unit.prototype.lowestLevel = function() {
    return $gameParty.lowestLevel();
};

Game_Unit.prototype.averageLevel = function() {
    return $gameParty.averageLevel();
};

Yanfly.CoreAI.Game_Unit_onBattleStart = Game_Unit.prototype.onBattleStart;
Game_Unit.prototype.onBattleStart = function() {
    Yanfly.CoreAI.Game_Unit_onBattleStart.call(this);
};

Game_Unit.prototype.aiElementRateKnown = function(target, elementId) {
    return true;
};
```

```
Game_Unit.prototype.aiRegisterElementRate = function(target, elementId) {
};

//=============================================================================
// Game_Party
//=============================================================================

Game_Party.prototype.lowestLevel = function() {
    return Math.min.apply(null, this.members().map(function(actor) {
        return actor.level;
    }));
};

Game_Party.prototype.averageLevel = function() {
    var level = 0;
    for (var i = 0; i < this.members().length; ++i) {
      var member = this.members()[i];
      if (member) level += member.level;
    }
    level /= this.members().length;
    return level;
};

//=============================================================================
// Game_Troop
//=============================================================================

Game_Troop.prototype.updateAIPatterns = function() {
    for (var i = 0; i < this.aliveMembers().length; ++i) {
      var member = this.aliveMembers()[i];
      if (member) member.setAIPattern();
    }
};
```

```
Yanfly.CoreAI.Game_Troop_setup = Game_Troop.prototype.setup;
Game_Troop.prototype.setup = function(troopId) {
    Yanfly.CoreAI.Game_Troop_setup.call(this, troopId);
    this._aiKnownElementRates = {};
};

Game_Troop.prototype.aiElementRateKnown = function(target, elementId) {
    if (target.isEnemy()) return true;
    if (!Yanfly.Param.CoreAIElementTest) return true;
    var index = target.index();
    if (this._aiKnownElementRates[index] === undefined) {
      this._aiKnownElementRates[index] = [];
    }
    return this._aiKnownElementRates[index].contains(elementId);
};

Game_Troop.prototype.aiRegisterElementRate = function(target, elementId) {
    if (!Yanfly.Param.CoreAIElementTest) return;
    var index = target.index();
    if (this._aiKnownElementRates[index] === undefined) {
      this._aiKnownElementRates[index] = [];
    }
    if (!this._aiKnownElementRates[index].contains(elementId)) {
      this._aiKnownElementRates[index].push(elementId);
    }
};

//=============================================================================
// Game_Action
//=============================================================================

Yanfly.CoreAI.Game_Action_apply = Game_Action.prototype.apply;
Game_Action.prototype.apply = function(target) {
    Yanfly.CoreAI.Game_Action_apply.call(this, target);
```

```
    this.aiRegisterElementRate(target);
};

Game_Action.prototype.aiRegisterElementRate = function(target) {
    if (this.item().damage.elementId < 0) return;
    var elementId = this.item().damage.elementId;
    if (this.subject().isActor()) {
      $gameParty.aiRegisterElementRate(target, elementId);
    } else {
      $gameTroop.aiRegisterElementRate(target, elementId);
    }
};

//=============================================================================
// AIManager
//=============================================================================

function AIManager() {
    throw new Error('This is a static class');
}

AIManager.setBattler = function(battler) {
    this._battler = battler;
    this._action = battler.action(0);
};

AIManager.battler = function() {
    return this._battler;
};

AIManager.action = function() {
    return this._action;
};
```

```
AIManager.isDecidedActionAI = function(line) {
    if (line.match(/[ ]*(.*):[ ](?:SKILL)[ ](\d+),[ ](.*)/i)) {
      this._origCondition =  String(RegExp.$1);
      var condition = String(RegExp.$1);
      this._aiSkillId = parseInt(RegExp.$2);
      this._aiTarget = String(RegExp.$3);
    } else if (line.match(/[ ]*(.*):[ ](?:SKILL)[ ](\d+)/i)) {
      this._origCondition =  String(RegExp.$1);
      var condition = String(RegExp.$1);
      this._aiSkillId = parseInt(RegExp.$2);
      this._aiTarget = 'RANDOM';
    } else if (line.match(/[ ]*(.*):[ ](.*),[ ](.*)/i)) {
      this._origCondition =  String(RegExp.$1);
      var condition = String(RegExp.$1);
      this._aiSkillId = Yanfly.SkillIdRef[String(RegExp.$2).toUpperCase()];
      this._aiTarget = String(RegExp.$3);
    } else if (line.match(/[ ]*(.*):[ ](.*)/i)) {
      this._origCondition = String(RegExp.$1);
      var condition = String(RegExp.$1);
      this._aiSkillId = Yanfly.SkillIdRef[String(RegExp.$2).toUpperCase()];
      this._aiTarget = 'RANDOM';
    }  else  {
      return false;
    }
    if (!this.initialCheck(this._aiSkillId)) return false;
    if (!this.meetCustomAIConditions(this._aiSkillId)) return false;
    this.action().setSkill(this._aiSkillId);
    if (!this.passAllAIConditions(condition)) return false;
    return true;
};

AIManager.getPatternSkills = function(array, patterns) {
    for (var i = 0; i < patterns.length; ++i) {
      var line = patterns[i];
```

```
      if (line.match(/[ ]*(.*):[ ](?:SKILL)[ ](\d+),[ ](.*)/i)) {
        var skillId = parseInt(RegExp.$2);
      } else if (line.match(/[ ]*(.*):[ ](?:SKILL)[ ](\d+)/i)) {
        var skillId = parseInt(RegExp.$2);
      } else if (line.match(/[ ]*(.*):[ ](.*),[ ](.*)/i)) {
        var skillId = Yanfly.SkillIdRef[String(RegExp.$2).toUpperCase()];
      } else if (line.match(/[ ]*(.*):[ ](.*)/i)) {
        var skillId = Yanfly.SkillIdRef[String(RegExp.$2).toUpperCase()];
      } else {
        continue;
      }
      if ($dataSkills[skillId]) array.push($dataSkills[skillId]);
    }
    return array;
};

AIManager.initialCheck = function(skillId) {
  if (!$dataSkills[skillId]) return false;
  if (!this.hasSkill(skillId)) return false;
  return this.battler().meetsSkillConditions($dataSkills[skillId]);
};

AIManager.hasSkill = function(skillId) {
    return this.battler().hasSkill(skillId);
};

AIManager.meetCustomAIConditions = function(skillId) {
  return true;
};

AIManager.getActionGroup = function() {
  var action = this.action();
  if (Imported.YEP_X_SelectionControl) action.setSelectionFilter(true);
  if (!action) return [];
```

```
  if (action.isForUser()) {
    var group = [this.battler()];
  } else if (action.isForDeadFriend()) {
    var group = action.friendsUnit().deadMembers();
  } else if (action.isForFriend()) {
    var group = action.friendsUnit().aliveMembers();
  } else if (action.isForOpponent()) {
    if (this.battler().aiConsiderTaunt()) {
      $gameTemp._tauntMode = true;
      $gameTemp._tauntAction = action;
      var group = action.opponentsUnit().tauntMembers();
      $gameTemp._tauntMode = false;
      $gameTemp._tauntAction = undefined;
    } else {
      var group = action.opponentsUnit().aliveMembers();
    }
  } else {
    var group = [];
  }
  if (this._setActionGroup !== undefined) {
    group = Yanfly.Util.getCommonElements(this._setActionGroup, group);
  }
  this._setActionGroup = group;
  return this._setActionGroup;
};

AIManager.setActionGroup = function(group) {
  this._setActionGroup = group;
};

AIManager.setProperTarget = function(group) {
    this.setActionGroup(group);
    var action = this.action();
    var randomTarget = group[Math.floor(Math.random() * group.length)];
```

```
    if (!randomTarget) return action.setTarget(0);
    if (group.length <= 0) return action.setTarget(randomTarget.index());
    var line = this._aiTarget.toUpperCase();
    if (line.match(/FIRST/i)) {
      action.setTarget(0);
    } else if (line.match(/USER/i)) {
      var index = group.indexOf();
      action.setTarget(action.subject().index());
    } else if (line.match(/HIGHEST[ ](.*)/i)) {
      var param = this.getParamId(String(RegExp.$1));
      if (param < 0) return action.setTarget(randomTarget.index());
      if (param === 8) return this.setHighestHpFlatTarget(group);
      if (param === 9) return this.setHighestMpFlatTarget(group);
      if (param === 10) return this.setHighestHpRateTarget(group);
      if (param === 11) return this.setHighestMpRateTarget(group);
      if (param === 12) return this.setHighestLevelTarget(group);
      if (param === 13) return this.setHighestMaxTpTarget(group);
      if (param === 14) return this.setHighestTpTarget(group);
      if (param > 15) return action.setTarget(randomTarget.index());
      this.setHighestParamTarget(group, param);
    } else if (line.match(/LOWEST[ ](.*)/i)) {
      var param = this.getParamId(String(RegExp.$1));
      if (param < 0) return action.setTarget(randomTarget.index());
      if (param === 8) return this.setLowestHpFlatTarget(group);
      if (param === 9) return this.setLowestMpFlatTarget(group);
      if (param === 10) return this.setLowestHpRateTarget(group);
      if (param === 11) return this.setLowestMpRateTarget(group);
      if (param === 12) return this.setLowestLevelTarget(group);
      if (param === 13) return this.setLowestMaxTpTarget(group);
      if (param === 14) return this.setLowestTpTarget(group);
      if (param > 15) return action.setTarget(randomTarget.index());
      this.setLowestParamTarget(group, param);
    } else {
      this.setRandomTarget(group);
```

```
    }
};


AIManager.getParamId = function(string) {
    string = string.toUpperCase()
    switch (string) {
    case 'MAXHP':
    case 'MAX HP':
      return 0;
      break;
    case 'MAXMP':
    case 'MAX MP':
     case 'MAXSP':
    case 'MAX SP':
       return 1;
      break;
    case 'ATK':
    case 'STR':
      return 2;
      break;
    case 'DEF':
      return 3;
      break;
    case 'MAT':
    case 'INT':
    case 'SPI':
      return 4;
      break;
    case 'MDF':
    case 'RES':
      return 5;
      break;
    case 'AGI':
    case 'SPD':
```

```
      return 6;
      break;
    case 'LUK':
      return 7;
      break;
    case 'HP':
      return 8;
      break;
    case 'MP':
    case 'SP':
      return 9;
      break;
    case 'HP%':
      return 10;
      break;
    case 'MP%':
    case 'SP%':
      return 11;
      break;
    case 'LEVEL':
    case 'LV':
    case 'LVL':
      return 12;
      break;
    case 'MAXTP':
      return 13;
      break;
    case 'TP':
      return 14;
      break;
    }
    return -1;
};
```

```
AIManager.setHighestHpRateTarget = function(group) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.hp / target.mhp > maintarget.hp / maintarget.mhp) {
        maintarget = target;
      }
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setHighestHpFlatTarget = function(group) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.hp > maintarget.hp) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setLowestHpRateTarget = function(group) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.hp / target.mhp < maintarget.hp / maintarget.mhp) {
        maintarget = target;
      }
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setLowestHpFlatTarget = function(group) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
```

```
      var target = group[i];
      if (target.hp < maintarget.hp) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setHighestMpRateTarget = function(group) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.mp / target.mmp > maintarget.mp / maintarget.mmp) {
        maintarget = target;
      }
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setHighestMpFlatTarget = function(group) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.mp > maintarget.mp) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setLowestMpRateTarget = function(group) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.mp / target.mmp < maintarget.mp / maintarget.mmp) {
        maintarget = target;
      }
    }
```

```
    this.action().setTarget(maintarget.index())
};

AIManager.setLowestMpFlatTarget = function(group) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.mp < maintarget.mp) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setHighestParamTarget = function(group, id) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.param(id) > maintarget.param(id)) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setLowestParamTarget = function(group, id) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.param(id) < maintarget.param(id)) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setHighestLevelTarget = function(group, id) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
```

```
      if (target.level > maintarget.level) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setLowestLevelTarget = function(group, id) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.level < maintarget.level) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setHighestMaxTpTarget = function(group, id) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.tp > maintarget.maxTp()) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setLowestMaxTpTarget = function(group, id) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.tp < maintarget.maxTp()) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setHighestTpTarget = function(group, id) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
```

```
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.tp > maintarget.tp) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setLowestTpTarget = function(group, id) {
    var maintarget = group[Math.floor(Math.random() * group.length)];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (target.tp < maintarget.tp) maintarget = target;
    }
    this.action().setTarget(maintarget.index())
};

AIManager.setRandomTarget = function(group) {
    var target = group[Math.floor(Math.random() * group.length)];
    this.action().setTarget(target.index())
};

AIManager.convertIntegerPercent = function(n) {
    n *= 0.01
    return String(n);
};

AIManager.elementRateMatch = function(target, elementId, type) {
    var rate = target.elementRate(elementId).toFixed(2);
    if (this.battler().isActor()) {
      if (!$gameParty.aiElementRateKnown(target, elementId)) return true;
    } else {
      if (!$gameTroop.aiElementRateKnown(target, elementId)) return true;
    }
    if (['NEUTRAL', 'NORMAL'].contains(type)) {
```

```
      return rate >= 0.90 && rate <= 1.10;
    } else if (['WEAK', 'WEAKNESS', 'VULNERABLE'].contains(type)) {
      return rate > 1.00;
    } else if (['RESIST', 'RESISTANT', 'STRONG'].contains(type)) {
      return rate < 1.00;
    } else if (['NULL', 'CANCEL', 'NO EFFECT'].contains(type)) {
      return rate === 0.00;
    } else if (['ABSORB', 'HEAL'].contains(type)) {
      return rate < 0.00;
    }
    return false;
};

AIManager.passAllAIConditions = function(line) {
  this._setActionGroup = undefined;
  var conditions = line.split('+++');
  if (conditions.length <= 0) return false;
  while (conditions.length > 0) {
    var condition = conditions.shift().trim();
    if (!this.passAIConditions(condition)) return false;
  }
  return true;
};

AIManager.passAIConditions = function(line) {
    // ALWAYS
    if (line.match(/ALWAYS/i)) {
      return this.conditionAlways();
    }
    // ELEMENT
    if (line.match(/ELEMENT[ ](.*)/i)) {
      return this.conditionElement();
    }
    // EVAL
```

```
    if (line.match(/EVAL[ ](.*)/i)) {
      var condition = String(RegExp.$1);
      return this.conditionEval(condition);
    }
    // GROUP ALIVE MEMBERS EVAL
    if (line.match(/(.*)[ ]ALIVE[ ]MEMBERS[ ](.*)/i)) {
      var members = String(RegExp.$1);
      var condition = String(RegExp.$2);
      return this.conditionGroupAlive(members, condition);
    }
    // GROUP DEAD MEMBERS EVAL
    if (line.match(/(.*)[ ]DEAD[ ]MEMBERS[ ](.*)/i)) {
      var members = String(RegExp.$1);
      var condition = String(RegExp.$2);
      return this.conditionGroupDead(members, condition);
    }
    // USER PARAM EVAL
    if (line.match(/USER[ ](.*)[ ]PARAM[ ](.*)/i)) {
      var paramId = this.getParamId(String(RegExp.$1));
      var condition = String(RegExp.$2);
      return this.conditionUserParamEval(paramId, condition);
    }
    // PARAM EVAL
    if (line.match(/(.*)[ ]PARAM[ ](.*)/i)) {
      var paramId = this.getParamId(String(RegExp.$1));
      var condition = String(RegExp.$2);
      return this.conditionParamEval(paramId, condition);
    }
    // PARTY LEVEL
    if (line.match(/(.*)[ ]PARTY[ ]LEVEL[ ](.*)/i)) {
      var type = String(RegExp.$1);
      var condition = String(RegExp.$2);
      return this.conditionPartyLevel(type, condition);
    }
```

```
    // RANDOM x%
    if (line.match(/RANDOM[ ](\d+)([%%])/i)) {
      return this.conditionRandom(parseFloat(RegExp.$1 * 0.01));
    }
    // STATE === X
    if (line.match(/STATE[ ]===[ ](.*)/i)) {
      return this.conditionStateHas(String(RegExp.$1));
    }
    // STATE !== X
    if (line.match(/STATE[ ]!==[ ](.*)/i)) {
      return this.conditionStateNot(String(RegExp.$1));
    }
    // SWITCH X case
    if (line.match(/SWITCH[ ](\d+)[ ](.*)/i)) {
      var switchId = parseInt(RegExp.$1);
      var value = String(RegExp.$2)
      return this.conditionSwitch(switchId, value);
    }
    // TURN EVAL
    if (line.match(/TURN[ ](.*)/i)) {
      return this.conditionTurnCount(String(RegExp.$1));
    }
    // VARIABLE X eval
    if (line.match(/VARIABLE[ ](\d+)[ ](.*)/i)) {
      var variableId = parseInt(RegExp.$1);
      var condition = String(RegExp.$2)
      return this.conditionVariable(variableId, condition);
    }
    return false;
};

AIManager.conditionAlways = function() {
    var group = this.getActionGroup();
    this.setProperTarget(group);
```

```
    return true;
};

AIManager.conditionElement = function() {
    var line = this._origCondition;
    if (line.match(/ELEMENT[ ](\d+)[ ](.*)/i)) {
      var elementId = parseInt(RegExp.$1);
      var type = String(RegExp.$2).toUpperCase();
    } else if (line.match(/ELEMENT[ ](.*)[ ](.*)/i)) {
      var elementId = Yanfly.ElementIdRef[String(RegExp.$1).toUpperCase()];
      var type = String(RegExp.$2).toUpperCase();
    }  else  {
      return false;
    }
    var group = this.getActionGroup();
    var validTargets = [];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (!target) continue;
      if (this.elementRateMatch(target, elementId, type)) {
        validTargets.push(target);
      }
    }
    if (validTargets.length <= 0) return false;
    this.setProperTarget(validTargets);
    return true;
};

AIManager.conditionEval = function(condition) {
    var action = this.action();
    var item = action.item();
    var user = this.battler();
    var s = $gameSwitches._data;
    var v = $gameVariables._data;
```

```
    try {
      if (!eval(condition)) return false;
    } catch (e) {
      Yanfly.Util.displayError(e, condition, 'A.I. EVAL ERROR');
      return false;
    }
    var group = this.getActionGroup();
    this.setProperTarget(group);
    return true;
};

AIManager.conditionGroupAlive = function(members, condition) {
    var action = this.action();
    var item  =  action.item();
    var user = this.battler();
    var s = $gameSwitches._data;
    var v = $gameVariables._data;
    members = members.toUpperCase();
    if (['TROOP', 'TROOPS', 'ENEMY', 'ENEMIES'].contains(members)) {
      members = $gameTroop.aliveMembers();
    } else if (['PARTY', 'PLAYER'].contains(members)) {
      members = $gameParty.aliveMembers();
    }  else  {
      return false;
    }
    if (members.length <= 0) return false;
    condition = 'members.length ' + condition;
    try {
      if (!eval(condition)) return false;
    } catch (e) {
      Yanfly.Util.displayError(e, condition, 'A.I. GROUP ALIVE ERROR');
      return false;
    }
    var group = this.getActionGroup();
```

```
    this.setProperTarget(group);
    return true;
};

AIManager.conditionGroupDead = function(members, condition) {
    var action = this.action();
    var item  =  action.item();
    var user = this.battler();
    var s = $gameSwitches._data;
    var v = $gameVariables._data;
    members = members.toUpperCase();
    if (['TROOP', 'TROOPS', 'ENEMY', 'ENEMIES'].contains(members)) {
      members = $gameTroop.deadMembers();
    } else if (['PARTY', 'PLAYER'].contains(members)) {
      members = $gameParty.deadMembers();
    }  else  {
      return false;
    }
    if (members.length <= 0) return false;
    condition = 'members.length ' + condition;
    try {
      if (!eval(condition)) return false;
    } catch (e) {
      Yanfly.Util.displayError(e, condition, 'A.I. GROUP DEAD ERROR');
      return false;
    }
    var group = this.getActionGroup();
    this.setProperTarget(group);
    return true;
};

AIManager.conditionPartyLevel = function(type, condition) {
    if (type.match(/HIGHEST/i)) {
      condition = '.highestLevel() ' + condition;
```

```
    } else if (type.match(/LOWEST/i)) {
      condition = '.lowestLevel() ' + condition;
    } else if (type.match(/AVERAGE/i)) {
      condition = '.averageLevel() ' + condition;
    }  else  {
      return false;
    }
    var action = this.action();
    var item  =  action.item();
    var user = this.battler();
    var s = $gameSwitches._data;
    var v = $gameVariables._data;
    if (action.isForFriend()) {
      condition = 'action.friendsUnit()' + condition;
    } else if (action.isForOpponent()) {
      condition = 'action.opponentsUnit()' + condition;
    }
    try {
      if (!eval(condition)) return false;
    } catch (e) {
      Yanfly.Util.displayError(e, condition, 'A.I. PARTY LEVEL ERROR');
      return false;
    }
    var group = this.getActionGroup();
    this.setProperTarget(group);
    return true;
};

AIManager.conditionUserParamEval = function(paramId, condition) {
    var action = this.action();
    var item = action.item();
    var user = this.battler();
    var s = $gameSwitches._data;
    var v = $gameVariables._data;
```

```
    condition = condition.replace(/(\d+)([%%])/g, function() {
      return this.convertIntegerPercent(parseInt(arguments[1]));
    }.bind(this));
    if (paramId < 0) return false;
    if (paramId >= 0 && paramId <= 7) {
      condition = 'user.param(paramId) ' + condition;
    } else if (paramId === 8) {
      condition = 'user.hp ' + condition;
    } else if (paramId === 9) {
      condition = 'user.mp ' + condition;
    } else if (paramId === 10) {
      condition = 'user.hp / user.mhp ' + condition;
    } else if (paramId === 11) {
      condition = 'user.mp / user.mmp ' + condition;
    } else if (paramId === 12) {
      condition = 'user.level ' + condition;
    }
    var group = this.getActionGroup();
    var validTargets = [];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (!target) continue;
      try {
        if (eval(condition)) validTargets.push(target);
      } catch (e) {
        Yanfly.Util.displayError(e, condition, 'A.I. USER PARAM ERROR')
      }
    }
    if (validTargets.length <= 0) return false;
    this.setProperTarget(validTargets);
    return true;
};

AIManager.conditionParamEval = function(paramId, condition) {
```

```
    var action = this.action();
    var item  =  action.item();
    var user = this.battler();
    var s = $gameSwitches._data;
    var v = $gameVariables._data;
    condition = condition.replace(/(\d+)([%%])/g, function() {
      return this.convertIntegerPercent(parseInt(arguments[1]));
    }.bind(this));
    if (paramId < 0) return false;
    if (paramId >= 0 && paramId <= 7) {
      condition = 'target.param(paramId) ' + condition;
    } else if (paramId === 8) {
      condition = 'target.hp ' + condition;
    } else if (paramId === 9) {
      condition = 'target.mp ' + condition;
    } else if (paramId === 10) {
      condition = 'target.hp / target.mhp ' + condition;
    } else if (paramId === 11) {
      condition = 'target.mp / target.mmp ' + condition;
    } else if (paramId === 12) {
      condition = 'target.level ' + condition;
    }
    var group = this.getActionGroup();
    var validTargets = [];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (!target) continue;
      try {
        if (eval(condition)) validTargets.push(target);
      } catch (e) {
        Yanfly.Util.displayError(e, condition, 'A.I. PARAM ERROR')
      }
    }
    if (validTargets.length <= 0) return false;
```

```
    this.setProperTarget(validTargets);
    return true;
};

AIManager.conditionRandom = function(rate) {
    if (Math.random() >= rate) return false;
    var group = this.getActionGroup();
    this.setProperTarget(group);
    return true;
};

AIManager.conditionStateHas = function(condition) {
    if (condition.match(/STATE[ ](\d+)/i)) {
      var stateId = parseInt(RegExp.$1);
    } else {
      var stateId = Yanfly.StateIdRef[condition.toUpperCase()];
      if (!stateId) return false;
    }
    if (!$dataStates[stateId]) return false;
    var group = this.getActionGroup();
    var validTargets = [];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (!target) continue;
      if (target.hasState(stateId)) validTargets.push(target);
    }
    if (validTargets.length <= 0) return false;
    this.setProperTarget(validTargets);
    return true;
};

AIManager.conditionStateNot = function(condition) {
    if (condition.match(/STATE[ ](\d+)/i)) {
      var stateId = parseInt(RegExp.$1);
```

```
    } else {
      var stateId = Yanfly.StateIdRef[condition.toUpperCase()];
      if (!stateId) return false;
    }
    if (!$dataStates[stateId]) return false;
    var group = this.getActionGroup();
    var validTargets = [];
    for (var i = 0; i < group.length; ++i) {
      var target = group[i];
      if (!target) continue;
      if (target.notState(stateId)) validTargets.push(target);
    }
    if (validTargets.length <= 0) return false;
    this.setProperTarget(validTargets);
    return true;
};

AIManager.conditionSwitch = function(switchId, value) {
    if (['ON', 'TRUE', 'YES'].contains(value.toUpperCase())) {
      value = true;
    } else if (['OFF', 'FALSE', 'NO'].contains(value.toUpperCase())) {
      value = false;
    }  else  {
      return false;
    }
    if ($gameSwitches.value(switchId) !== value) return false;
    var group = this.getActionGroup();
    this.setProperTarget(group);
    return true;
};

if (!Yanfly.Param.CoreAIDynTurnCnt) {

AIManager.conditionTurnCount = function(condition) {
```

```
    var action = this.action();
    var item  =  action.item();
    var user = this.battler();
    var s = $gameSwitches._data;
    var v = $gameVariables._data;
    if  (Imported.YEP_BattleEngineCore)  {
      condition = 'user.turnCount() ' + condition;
    } else {
      condition = '$gameTroop.turnCount() ' + condition;
    }
    try {
      if (!eval(condition)) return false;
    } catch (e) {
      Yanfly.Util.displayError(e, condition, 'A.I. TURN COUNT ERROR');
      return false;
    }
    var group = this.getActionGroup();
    this.setProperTarget(group);
    return true;
};

} else {
// Alternative provided by Talonos

AIManager.conditionTurnCount = function(condition) {
    var action = this.action();
    var item  =  action.item();
    var user = this.battler();
    var s = $gameSwitches._data;
    var v = $gameVariables._data;
    if (Imported.YEP_BattleEngineCore) {
      if (BattleManager._phase === "input" && BattleManager.isTurnBased()) {
        condition = '(user.turnCount() + 1) ' + condition;
      } else {
```

```
        condition = 'user.turnCount() ' + condition;
      }
    } else {
      if (BattleManager._phase === "input") {
        condition = '($gameTroop.turnCount() + 1) ' + condition;
      } else {
        condition = '$gameTroop.turnCount() ' + condition;
      }
    }
    try {
      if (!eval(condition)) return false;
    } catch (e) {
      Yanfly.Util.displayError(e, condition, 'A.I. TURN COUNT ERROR');
      return false;
    }
    var group = this.getActionGroup();
    this.setProperTarget(group);
    return true;
};
} // Yanfly.Param.CoreAIDynamic
AIManager.conditionVariable = function(variableId, condition) {
    var action = this.action();
    var item  =  action.item();
    var user = this.battler();
    var s = $gameSwitches._data;
    var v = $gameVariables._data;
    condition = '$gameVariables.value(' + variableId + ') ' + condition;
    try {
      if (!eval(condition)) return false;
    } catch (e) {
      Yanfly.Util.displayError(e, condition, 'A.I. VARIABLE ERROR');
      return false;
    }
    var group = this.getActionGroup();
```

```
    this.setProperTarget(group);
    return true;
};
//=============================================================================
// Utilities
//=============================================================================
Yanfly.Util = Yanfly.Util || {};
Yanfly.Util.displayError = function(e, code, message) {
  console.log(message);
  console.log(code || 'NON-EXISTENT');
  console.error(e);
  if (Utils.RPGMAKER_VERSION && Utils.RPGMAKER_VERSION >= "1.6.0") return;
  if (Utils.isNwjs() && Utils.isOptionValid('test')) {
    if (!require('nw.gui').Window.get().isDevToolsOpen()) {
      require('nw.gui').Window.get().showDevTools();
    }
  }
};
Yanfly.Util.getCommonElements = function(array1, array2) {
  var elements = [];
  var length = array1.length;
  for (var i = 0; i < length; ++i) {
    var element = array1[i];
    if (array2.contains(element)) elements.push(element);
  }
  return elements;
};
```

LEMBAR PERBAIKAN SKRIPSI

"APLIKASI GAME ANDROID EDUKASI KEPAHLAWAN SULTAN HASANUDDIN"

OLEH:

AHMAD SETIADI
D421 14 506

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 9 JUNI 2021.
Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

	Nama	Tanda Tagan
Ketua	Dr. Eng. Zulkifli Tahir, S.T., M.Sc.,	
Sekretaris	Elly Warni, S.T., M.T	
Anggota	DR.Indrabayu Amirullah, S.T., M.T, M.Bus sys	
	DR.Eng Dewiani, S.T., M.T	

Persetujuan perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tagan
I	Dr. Eng. Zulkifli Tahir, S.T., M.Sc.,	
II	Elly Warni, S.T., M.T	