Prison of Carbon

Refleksje na temat procesu tworzenia gry i projektowania

Dla koła naukowego RPG

Jakub „SceNtriC” Rojek

@SceNtriC

scentric@gmail.com

Adżenda for tudej

- Prison of Carbon – co to?
- Refleksje projektowe
- Refleksje techniczne
- Refleksje „biznesowe”

Refleksje techniczne

Cost Off-the-Shelf (COTS)

- Eclipse Juno + ADK + Logcat
- LibGDX (http://libgdx.badlogicgames.com/)
- Java + OpenGL (obudowany)

LibGDX

- Możliwe jednoczesne tworzenie kodu na Androida, iOS, HTML5 i Desktop
- Prosta darmowa biblioteka, dużo tutoriali
- Tak naprawdę wrapper na OpenGL oraz interfejs mobilny
- Wiele ułatwień (m.in. obsługa pamięci)
- Niezły tutorial, kiepska dokumentacja
- Alternatywa: PlayN

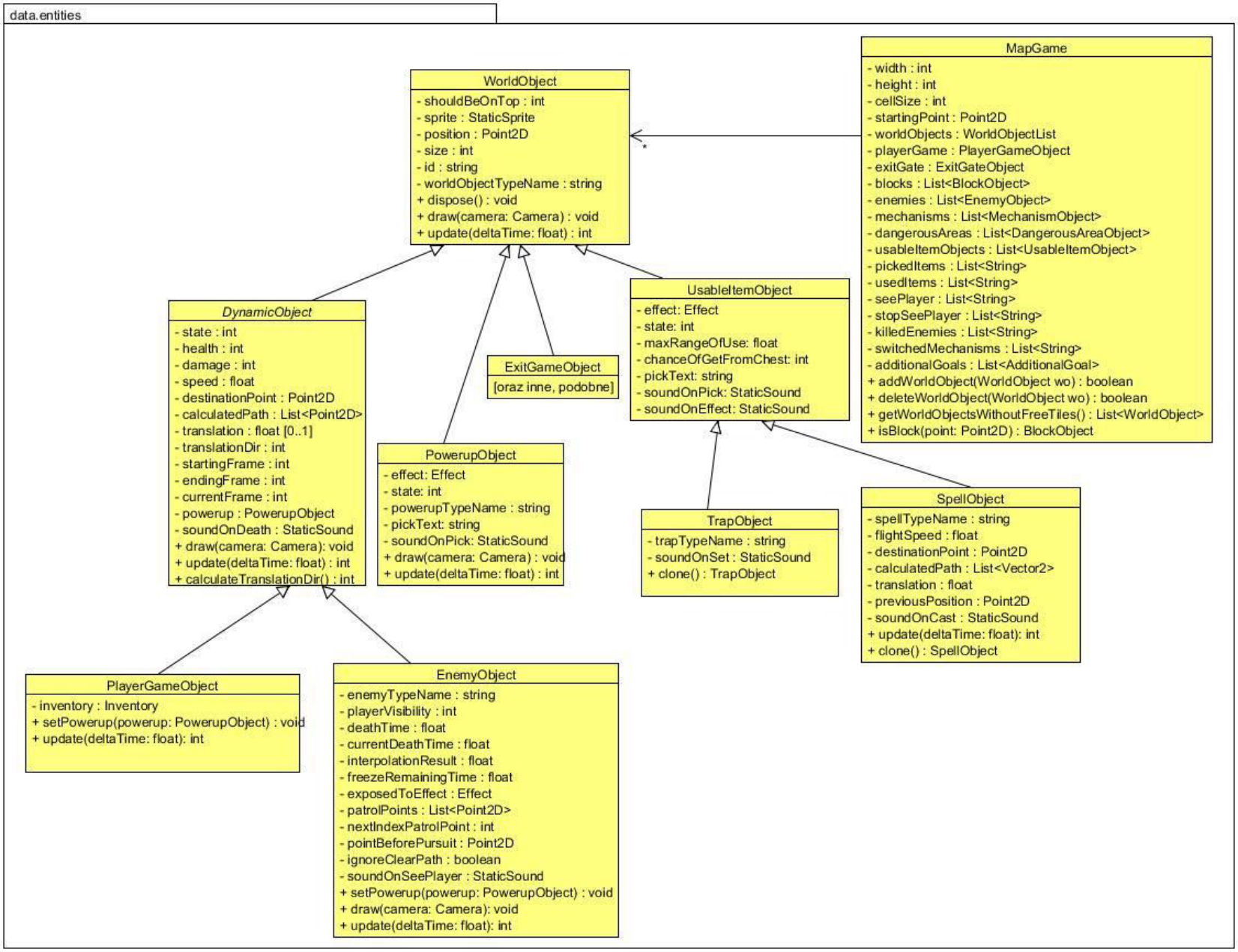
Struktura projektu

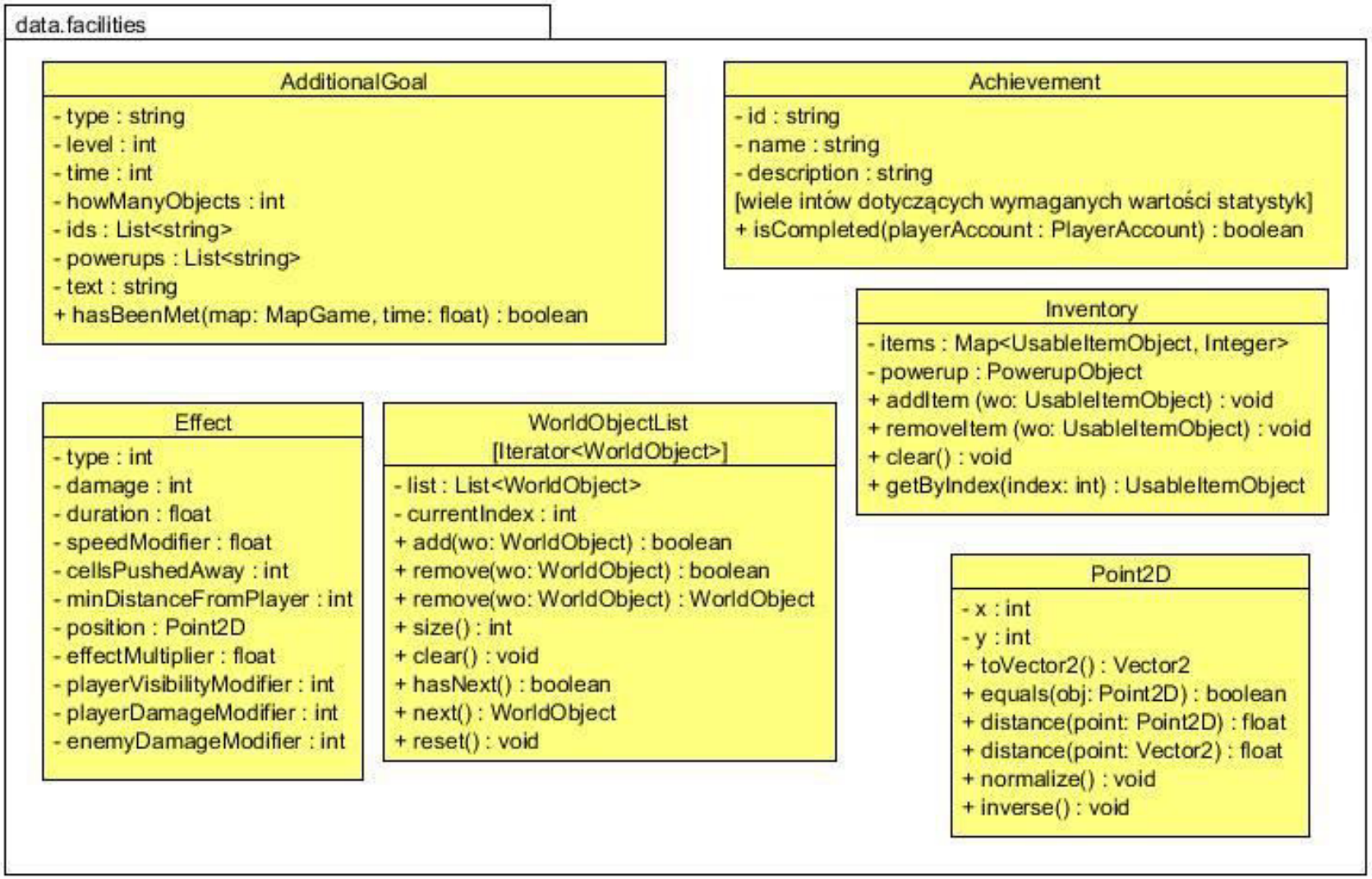
- Starter na Androida (jedynie konfiguracja)
- Główna klasa Javy (PrisonOfCarbon)

- Core (m.in. operacje matematyczne)
- Data (encje, loadery, managery)
- Logic (logika obiektów, mapy, obsługa kolizji)
- View (wyświetlanie mapy, GUI)

Ogólne założenia projektu kodu

- Obiektowość (ale nie do przesady)
- Podział na warstwy
- Brak strachu przed singletonami, choć mają swoje wady
- Pozycja obiektów wyrażona jest w wartościach całkowitych
- Wszystkie obiekty wywodzą się z jednej klasy


data.entities
MapGame
- width : int
- height : int
- cellSize : int
- startingPoint : Point2D
- worldObjects : WorldObjectList
- playerGame : PlayerGameObject
- exitGate : ExitGateObject
- blocks : List<BlockObject>
- enemies : List<EnemyObject>
- mechanisms : List<MechanismObject>
- dangerousAreas : List<DangerousAreaObject>
- usableItemObjects : List<UsableItemObject>
- pickedItems : List<String>
- usedItems : List<String>
- seePlayer : List<String>
- stopSeePlayer : List<String>
- killedEnemies : List<String>
- switchedMechanisms : List<String>
- additionalGoals : List<AdditionalGoal>
+ addWorldObject(WorldObject wo) : boolean
+ deleteWorldObject(WorldObject wo) : boolean
+ getWorldObjectsWithoutFreeTiles() : List<WorldObject>
+ isBlock(point: Point2D) : BlockObject
WorldObject
- shouldBeOnTop : int
- sprite : StaticSprite
- position : Point2D
- size : int
- id : string
- worldObjectTypeName : string
+ dispose() : void
+ draw(camera: Camera) : void
+ update(deltaTime: float) : int
*
DynamicObject
- state : int
- health : int
- damage : int
- speed : float
- destinationPoint : Point2D
- calculatedPath : List<Point2D>
- translation : float [0..1]
- translationDir : int
- startingFrame : int
- endingFrame : int
- currentFrame : int
- powerup : PowerupObject
- soundOnDeath : StaticSound
+ draw(camera: Camera): void
+ update(deltaTime: float) : int
+ calculateTranslationDir() : int
ExitGameObject
[oraz inne, podobne]
UsableItemObject
- effect: Effect
- state: int
- maxRangeOfUse: float
- chanceOfGetFromChest: int
- pickText: string
- soundOnPick: StaticSound
- soundOnEffect: StaticSound
PowerupObject
- effect: Effect
- state: int
- powerupTypeName : string
- pickText: string
- soundOnPick: StaticSound
+ draw(camera: Camera) : void
+ update(deltaTime: float) : int
TrapObject
- trapTypeName : string
- soundOnSet : StaticSound
+ clone() : TrapObject
SpellObject
- spellTypeName : string
- flightSpeed : float
- destinationPoint : Point2D
- calculatedPath : List<Vector2>
- translation : float
- previousPosition : Point2D
- soundOnCast : StaticSound
+ update(deltaTime: float): int
+ clone() : SpellObject
PlayerGameObject
- inventory : Inventory
+ setPowerup(powerup: PowerupObject) : void
+ update(deltaTime: float): int
EnemyObject
- enemyTypeName : string
- playerVisibility : int
- deathTime : float
- currentDeathTime : float
- interpolationResult : float
- freezeRemainingTime : float
- exposedToEffect : Effect
- patrolPoints : List<Point2D>
- nextIndexPatrolPoint : int
- pointBeforePursuit : Point2D
- ignoreClearPath : boolean
- soundOnSeePlayer : StaticSound
+ setPowerup(powerup: PowerupObject) : void
+ draw(camera: Camera): void
+ update(deltaTime: float): int


data.facilities
AdditionalGoal
- type : string
- level : int
- time : int
- howManyObjects : int
- ids : List<string>
- powerups : List<string>
- text : string
+ hasBeenMet(map: MapGame, time: float) : boolean
Achievement
- id : string
- name : string
- description : string
[wiele intów dotyczących wymaganych wartości statystyk]
+ isCompleted(playerAccount : PlayerAccount) : boolean
Inventory
- items : Map<UsableItemObject, Integer>
- powerup : PowerupObject
+ addItem (wo: UsableItemObject) : void
+ removeItem (wo: UsableItemObject) : void
+ clear() : void
+ getByIndex(index: int) : UsableItemObject
Effect
- type : int
- damage : int
- duration : float
- speedModifier : float
- cellsPushedAway : int
- minDistanceFromPlayer : int
- position : Point2D
- effectMultiplier : float
- playerVisibilityModifier : int
- playerDamageModifier : int
- enemyDamageModifier : int
WorldObjectList
[Iterator<WorldObject>]
- list : List<WorldObject>
- currentIndex : int
+ add(wo: WorldObject) : boolean
+ remove(wo: WorldObject) : boolean
+ remove(wo: WorldObject) : WorldObject
+ size() : int
+ clear() : void
+ hasNext() : boolean
+ next() : WorldObject
+ reset() : void
Point2D
- x : int
- y : int
+ toVector2() : Vector2
+ equals(obj: Point2D) : boolean
+ distance(point: Point2D) : float
+ distance(point: Vector2) : float
+ normalize() : void
+ inverse() : void

Encje

- Wszystkie obiekty poza mapą wywodzą się z klasy WorldObject, niektóre mają dodatkowe klasy bazowe
- Typy obiektów są identyfikowane prostą nazwą klasy, co jest wygodne, ale ciut wolniejsze niż mechanizmy Javy czy wartości numeryczne

Pozycja obiektów

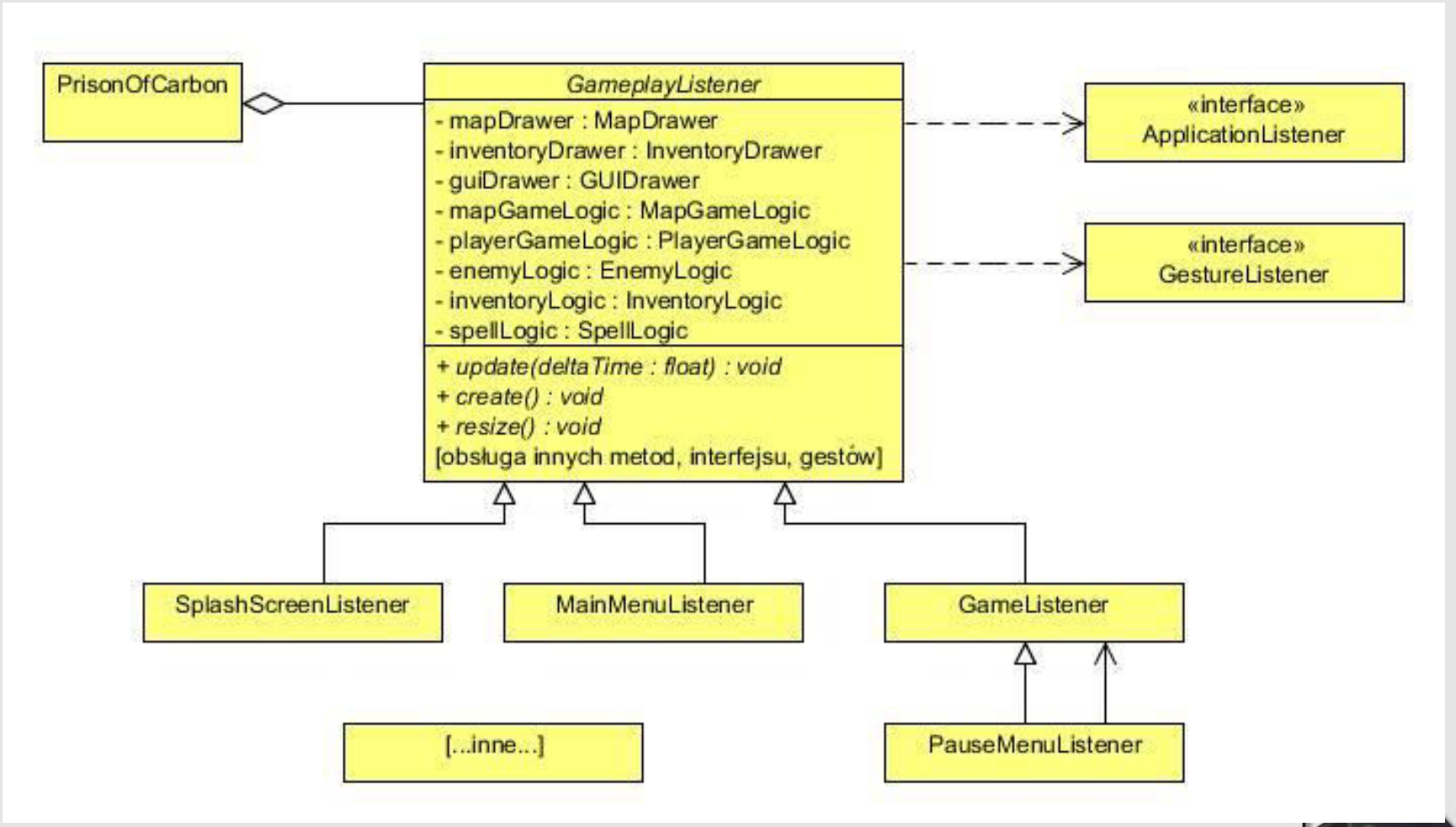
- Przyjęto liczby całkowite jako współrzędne
- A co z przejściami z pola na pole?
- Decyzja o braku refaktoryzacji – błąd
- Dodatkowe zmienne translation (od 0 do 1) oraz translationDir (kierunek i zwrot ruchu)
- Wiele problemów przy obliczaniu ścieżek, animacji, zmianach kierunku (czary)

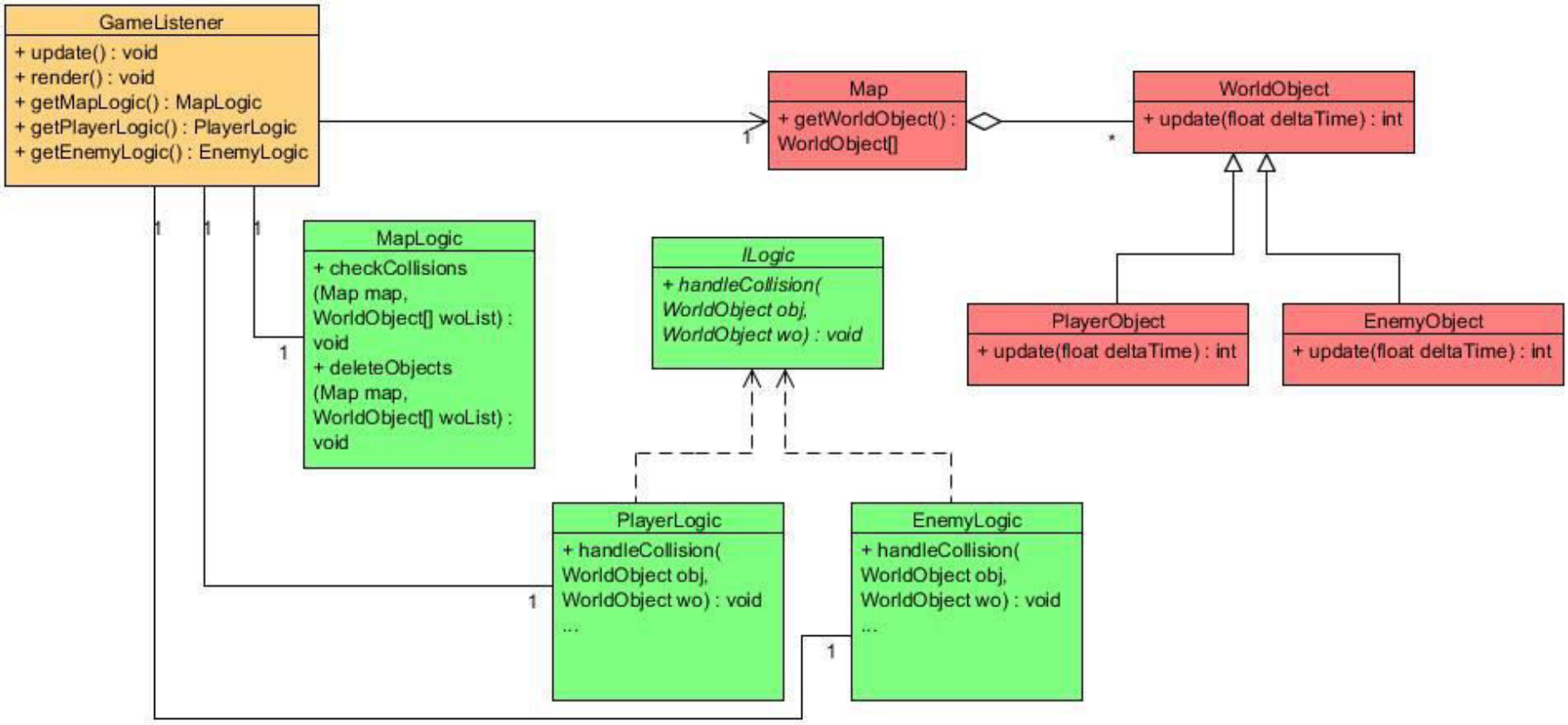
Wczytywanie zasobów

- Własne kontenery zasobów graficznych i dźwiękowych przechowujący pod kluczami zasoby wczytane loaderami z XML
- Powolny i problematyczny, singleton
- Dostępny AssetManager w LibGDX
- Polecane uzupełnienie go o dostęp pod kluczami (zwykle równe nazwie obiektów) oraz opakowanie w potrzebne rzeczy

Implementacja osiągnięć

- Paskudnie (nie)zaprojektowane
- Skompletowanie osiągnięcia polega na osiągnięciu statystyk większych lub równych wymaganym przez osiągnięcie
- Kilkadziesiąt różnych parametrów
- Sprawdzanie osiągnięć po każdej znaczącej akcji w grze (informacja od logiki), drabinka ifów


PrisonOfCarbon
GameplayListener
- mapDrawer : MapDrawer
- inventoryDrawer : InventoryDrawer
- guiDrawer : GUIDrawer
- mapGameLogic : MapGameLogic
- playerGameLogic : PlayerGameLogic
- enemyLogic : EnemyLogic
- inventoryLogic : InventoryLogic
- spellLogic : SpellLogic
+ update(deltaTime : float) : void
+ create() : void
+ resize() : void
[obsługa innych metod, interfejsu, gestów]
«interface»
ApplicationListener
«interface»
GestureListener
SplashScreenListener
MainMenuListener
GameListener
[...inne...]
PauseMenuListener



http://pseudodev.blogspot.com/2013/06/prosty-system-interakcji-pomiedzy.html

Założenia logiki (1/3)

- Osobne klasy dla trybów rozgrywki (referencja na aktualny tryb w PrisonOfCarbon – wygodne rozwiązanie, kwestia przepięcia referencji)
- Dwa tryby „wieczne” – MainMenu i Game
- Każdy tryb nadpisuje metody create, update, render i inne
- Tryby mają wskaźniki na poszczególne partie logiki
- Trzeba uważać przy obsłudze interfejsu – kwestia przycisków systemowych

Założenia logiki (2/3)

- Każda klasa logiki implementuje interfejs ILogic (PlayerGameLogic, MapLogic, …)
- Każda klasa ma metody modyfikujące parametry obiektów i obsługę kolizji
- MapLogic sprawdza czy uaktualnienie jakiegoś obiektu nie zwróciło nowej pozycji i sprawdzane są kolizje (nieefektywne), a później wywoływane handleCollision

Założenia logiki (3/3)

- handleCollision odpalają poszczególne metody w logice
- Warto zadbać o ograniczenie przeszukiwania interakcji pomiędzy obiektami (gdy to możliwe)
- Czy przesunięcie się obiektu to jedyny moment, kiedy trzeba sprawdzić kolizję?
- Wiele operacji w różnych miejscach (zadawanie obrażeń) – duplikacja a odseparowanie
- MapLogic – jedyna logika posiadająca swój obiekt (mapę)

Wyszukiwanie ścieżki

- Duże ułatwienie – kafelki oraz ruch możliwy maksymalnie w cztery strony
- http://www.policyalmanac.org/games/aStarTutorial_pl.htm
- Lekkie modyfikacje co do preferencji równoważnych pól (na ostatnim miejscu bramy wyjściowe i obszary niebezpieczne)
- Różne testy ścieżek patrolowych potworów ujawniały kolejne błędy, łatane dodatkowymi parametrami – automatyczne chodzenie postaci musi być bardzo dobrze zaprojektowane i zweryfikowane na „sucho”
- Inny problem – badanie pól pomiędzy dwoma polami

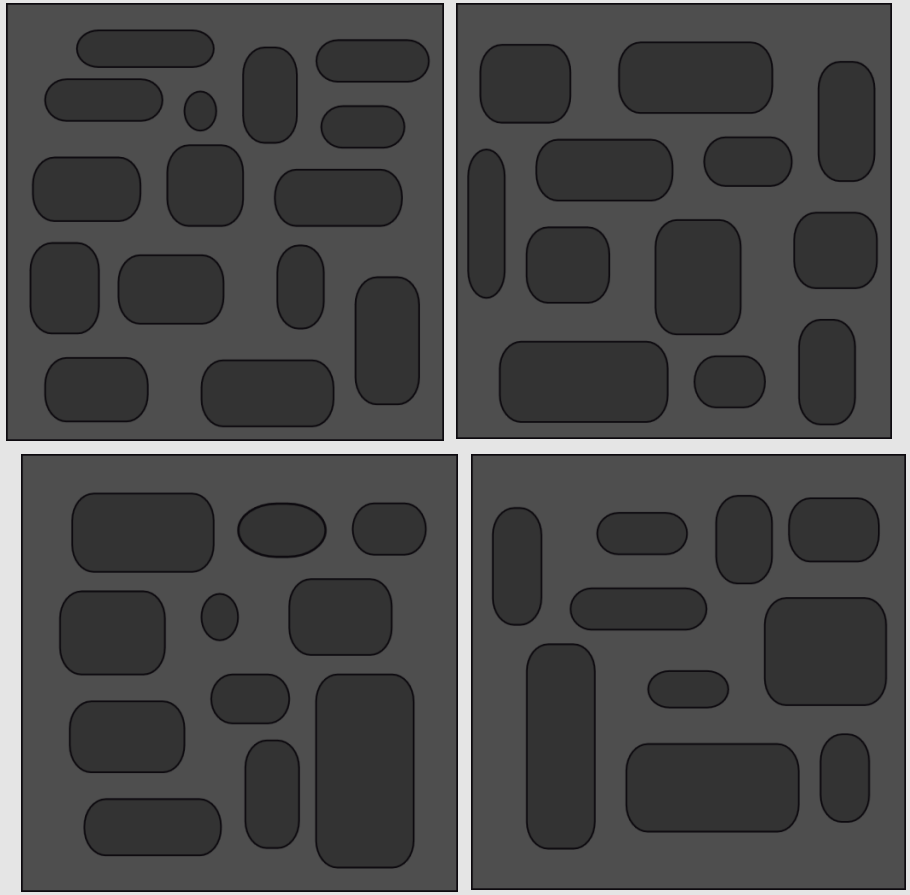
Wyświetlanie

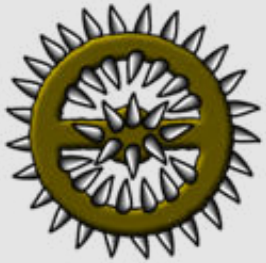
- Korzystanie z danych obiektów i logiki
- Pozycje elementów na ekranie „stałe”, parametryzowane przez wymiary ekranu
- Obszar przeznaczony na dany element (mapa, elementy HUDa) zaszyte w logice (zaburzenie architektury)
- Brak założenia internacjonalizacji – obecnie ciężko przetłumaczyć Prison of Carbon
- Postarać się o lepszą czcionkę

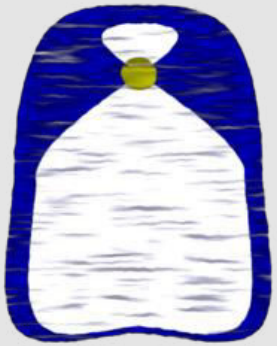
Grafika (1/3)

- Przy okazji GDD została opracowana wstępna lista potrzebnych zasobów (gfx i audio)
- Grafika wektorowa (Inkscape)
- http://2dgameartforprogrammers.blogspot.com/
- Kiepska jakość ze względu na kiepskiego wykonawcę
- Rozmiar grafik ma ogromny wpływ na rozmiar aplikacji i prędkość wczytywania – ograniczać, gdy tylko to możliwe
- Styl rysunkowy – wyraźne kontury, jednolite barwy
- **Grafika musi być spójna i utrzymana w jednakowym stylu**

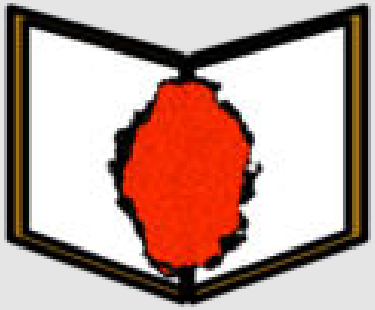
Grafika (2/3)

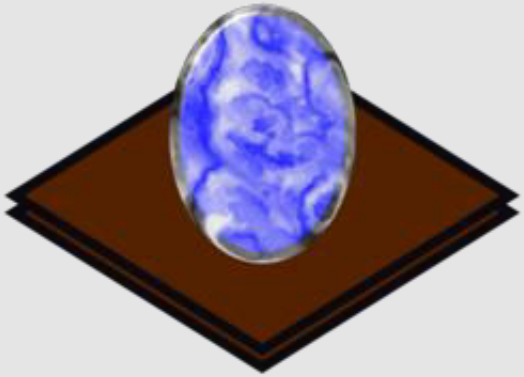






Grafika (3/3)

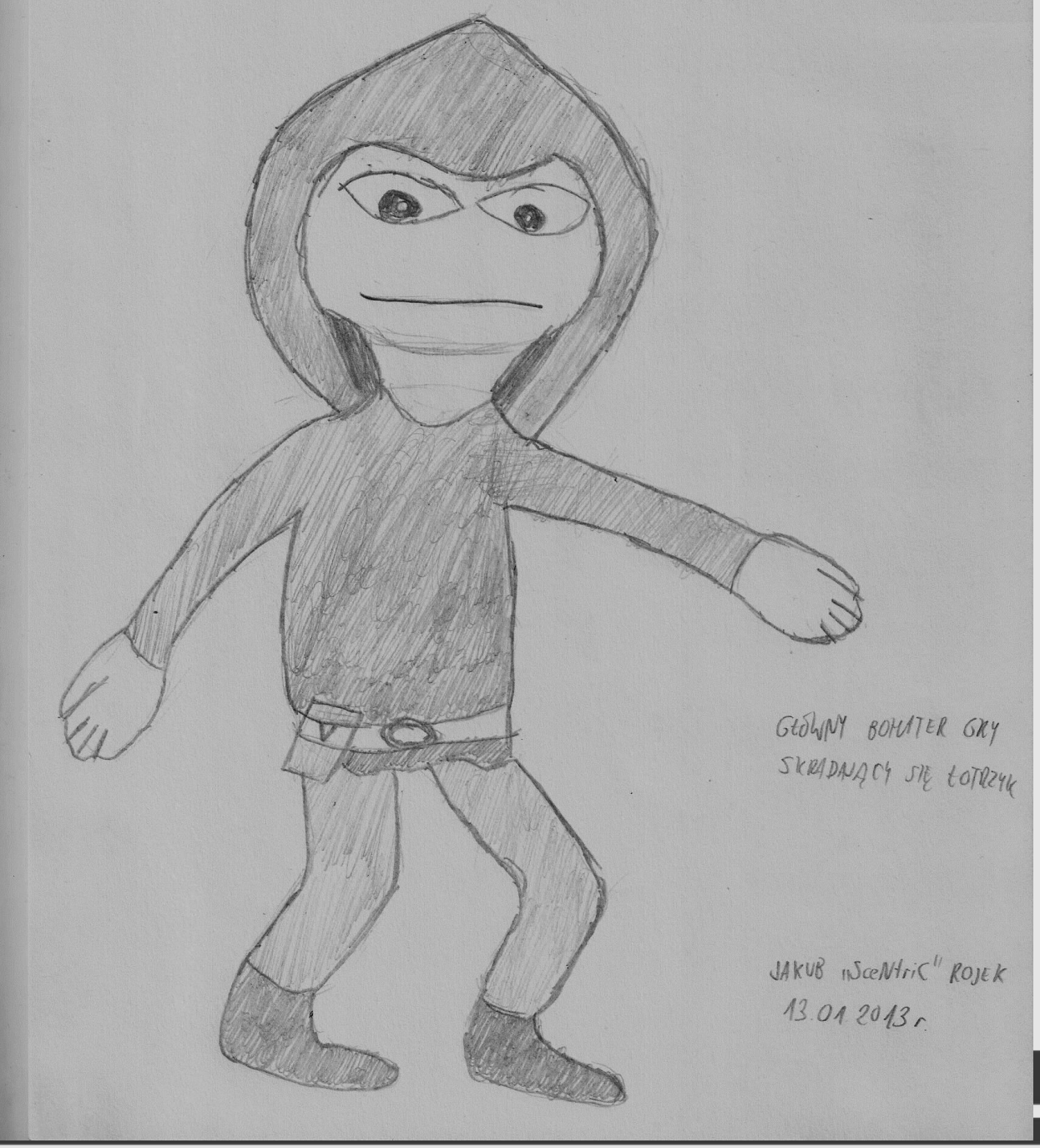


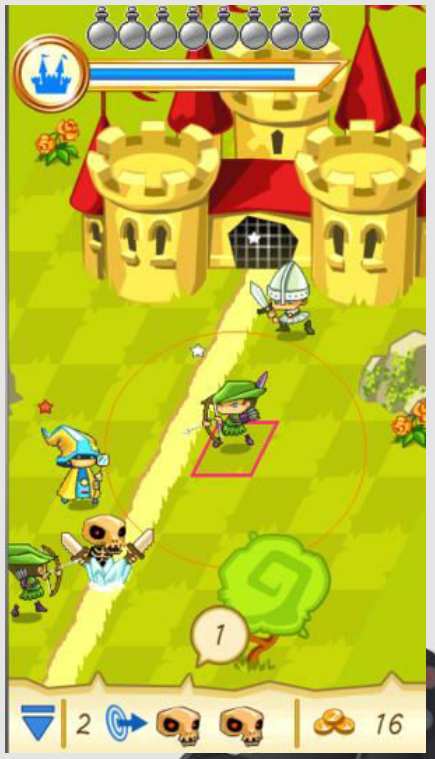


Grafiki (anty)koncepcyjne (1/5)

- Gracz oraz przeciwnicy mają różne klatki w zależności od zwrotu poruszania się oraz animacji chodu
- Warto szkicować dynamiczne obiekty, później łatwiejsze rysowanie
- Dobry materiał do pokazywania na Twitterze i innych serwisach
- Narysowanie postaci, a później tworzenie osobnych plików z lekkimi przesunięciami

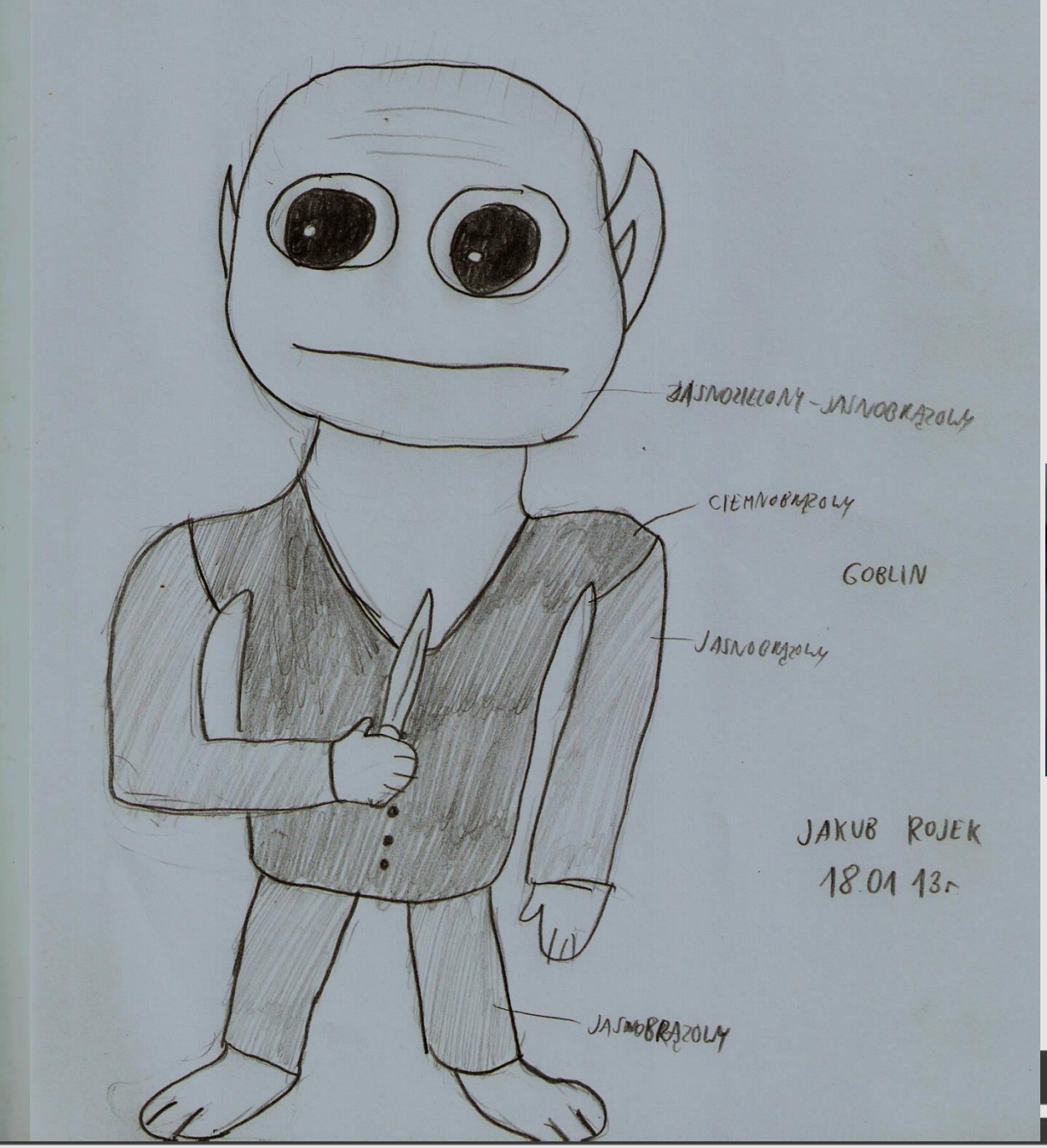
Grafiki (anty)koncepcyjne (2/5)





http://game400.com/wp-content/uploads/2013/03/fantasy1.png

Grafiki (anty)koncepcyjne (3/5)

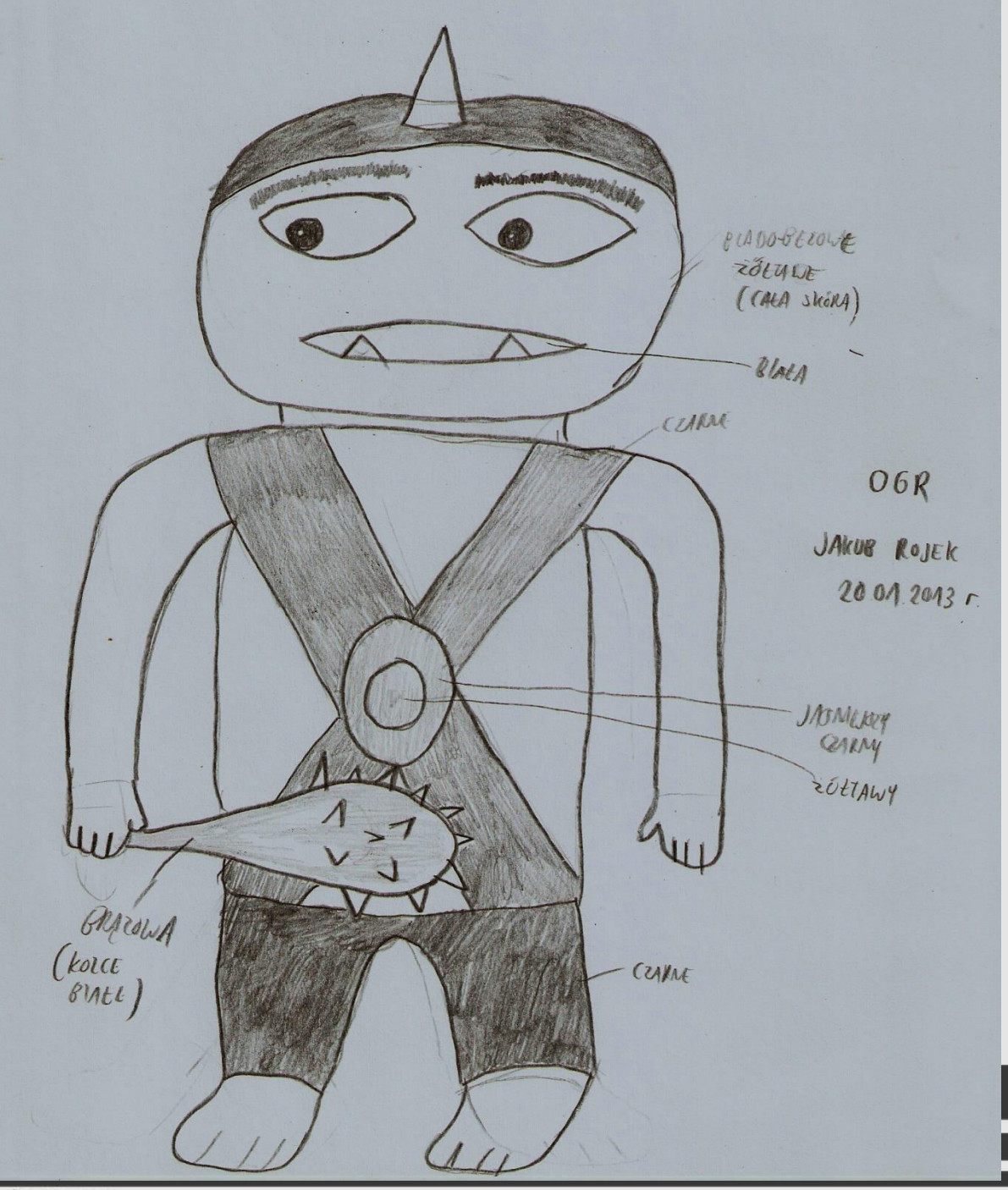


http://static1.wikia.nocookie.net/__cb20120731111358/harrypotter/pl/images/9/9e/Zgredek.jpg

Grafiki (anty)koncepcyjne (4/5)

http://static4.wikia.nocookie.net/__cb20090919170056/pacman/images/2/2b/Clydeeghost.png

Grafiki (anty)koncep cyjne (5/5)



http://upload.wikimedia.org/wikipedia/en/6/65/Gruntz_Coverart.png

Muzyka i dźwięk

- http://freesound.org/ - niezła baza darmowych dźwięków
- Wszystkie efekty dźwiękowe na CC0
- Często konieczność korekty w Audacity
- Stosunkowo łatwo znaleźć dźwięk, trudniej tło muzyczne
- Trzeba tak zaprojektować klasy, aby LibGDX nie przestawał odgrywać muzyki w tle przy zmianie okien np. w menu

Podsumowanie

- LibGDX jest niezłym wyborem na grę multiplatformową
- Warto dobrze przemyśleć architekturę aplikacji
- Można stosować podejście obiektowe (ale czy jest to najlepsze?)
- O spójność grafiki jest trudniej niż dźwięków

Dziękuję za uwagę!