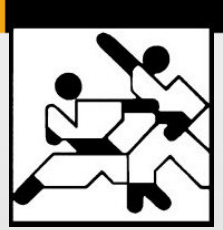
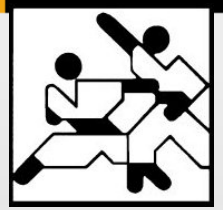


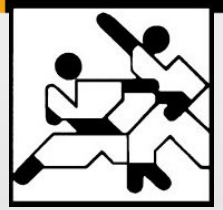
Techniki programowania gier

Dariusz Maciejewski



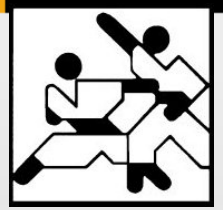
Plan prezentacji

- automatyczne singletony
- zarządzanie uchwytami do zasobów
- efektywne renderowanie z wykorzystaniem tablic wierzchołków
- animacja szkieletowa
- moja praca magisterska



Automatyczne singletony

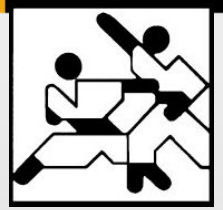
- co to jest singleton?
- dlaczego warto ich używać?
- singleton vs. globalny obiekt
- rozwiązanie ”szkolne”
- rozwiązanie alternatywne...
- ... z wykorzystaniem szablonów



Automatyczne singletony

rozwiązanie ”szkolne”:

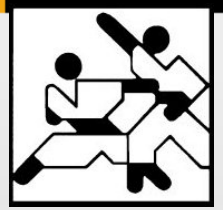
```
class Singleton
{
  private:
    Singleton() {}
   ~Singleton() {}
  public:
    Singleton &GetSingleton()
    {
      static Singleton s_Singleton;
      return s_Singleton;
    }
}

#define g_Singleton Singleton::GetSingleton()
```



Automatyczne singletony

alternatywne podejście:

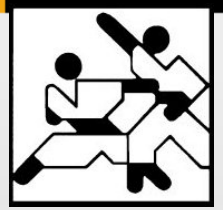
```
class Singleton {
  private:
    static Singleton* ms_Singleton;
  public:
    Singleton() {
      assert( !ms_Singleton );
      ms_Singleton = this;
    }
   ~Singleton() {
      assert( ms_Singleton ); ms_Singleton = 0;
    }
    Singleton &GetSingleton() {
      assert( ms_Singleton );
      return *ms_Singleton;
    } }
```



Automatyczne singletony

szablon klasy Singleton:

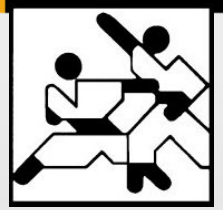
```
template <typename T> class Singleton {
  private:
    static T* ms_Singleton;
  public:
    Singleton() {
      assert( !ms_Singleton );
      int offset = (int)(T*)1
                   -(int)(Singleton<T>*)(T*)1;
      ms_Singleton = (T*)((int)this + offset);
    }
    /* ... */
};

template <typename T>
    T* Singleton<T>::ms_Singleton = 0;
```



Automatyczne singletony

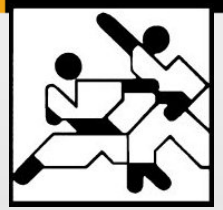
...i jego zastosowanie:

```
class TextMgr : public Singleton <TextMgr> {
  public:
    Texture* GetTexture( const char* name );
    /* ... */
};

#define g_TextMgr TextMgr::GetSingleton()

void Example( void )
{
  Texture* stone = TextMgr::GetSingleton().
                          GetTexture(„stone”);
  Texture* wood = g_TextMgr.GetTexture(„wood”);
}
```



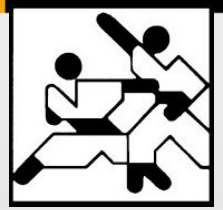
Uchwyty do zasobów

- zarządzanie zasobami w grach
  - tekstury
  - czcionki
  - postacie
  - dźwięki
  - ...
- wady wykorzystywania wskaźników
- zastosowanie uchwytów



Uchwyty do zasobów

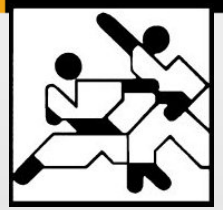
- zarządzanie zasobami w grach
- wady wykorzystywania wskaźników
  - czy dane można już bezpiecznie usunąć?
  - brak możliwości kontroli ważności wskaźnika
  - problem z zapisem/odczytem stanu gry
- zastosowanie uchwytów



Uchwyty do zasobów

- zastosowanie uchwytów
  - dodatkowa warstwa abstrakcji
  - prostota = szybkość

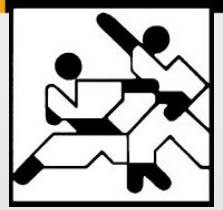
```
class Handle {
  union {
    struct {
      unsigned m_Index : 16; // indeks tablicy
      unsigned m_Magic : 16; // magiczna liczba
    };
    unsigned int m_Handle;
  };
  /* ... */
};
```



Uchwyty do zasobów

zarządca uchwytów:

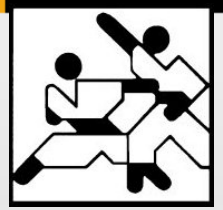
```
template <typename DATA> class HandleMgr {
  private:
    typedef std::vector <DATA>     DataVec;
    typedef std::vector <unsigned> MagicVec;
    typedef std::vector <unsigned> FreeVec;
    DataVec  m_Data;
    MagicVec m_Magic;
    FreeVec  m_FreeSlots;
  public:
    DATA          *Acquire( Handle& handle );
    void           Release( Handle handle );
    DATA          *Dereference( Handle handle );
    unsigned int   GetUsedHandleCount() const;
    bool           HasUsedHandles() const;
};
```



Uchwyty do zasobów

...i jego zastosowanie:

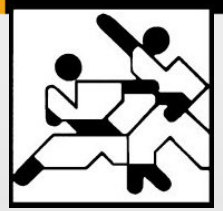
```
class TextMgr : public Singleton<TextMgr>  {
  private:
    class Texture { /* ... */ };
    HandleMgr<Texture> m_Textures;
    // name to handle std::map for searching
    NameIndex          m_NameIndex;

  public:
    Handle GetTexture    ( const char* name );
    void   DeleteTexture ( Handle htex );
    // Some operations on texture
    int    GetWidth      ( Handle htex ) const;
    int    GetHeight     ( Handle htex ) const;
    void   BindTexture   ( Handle htex ) const;
};
```



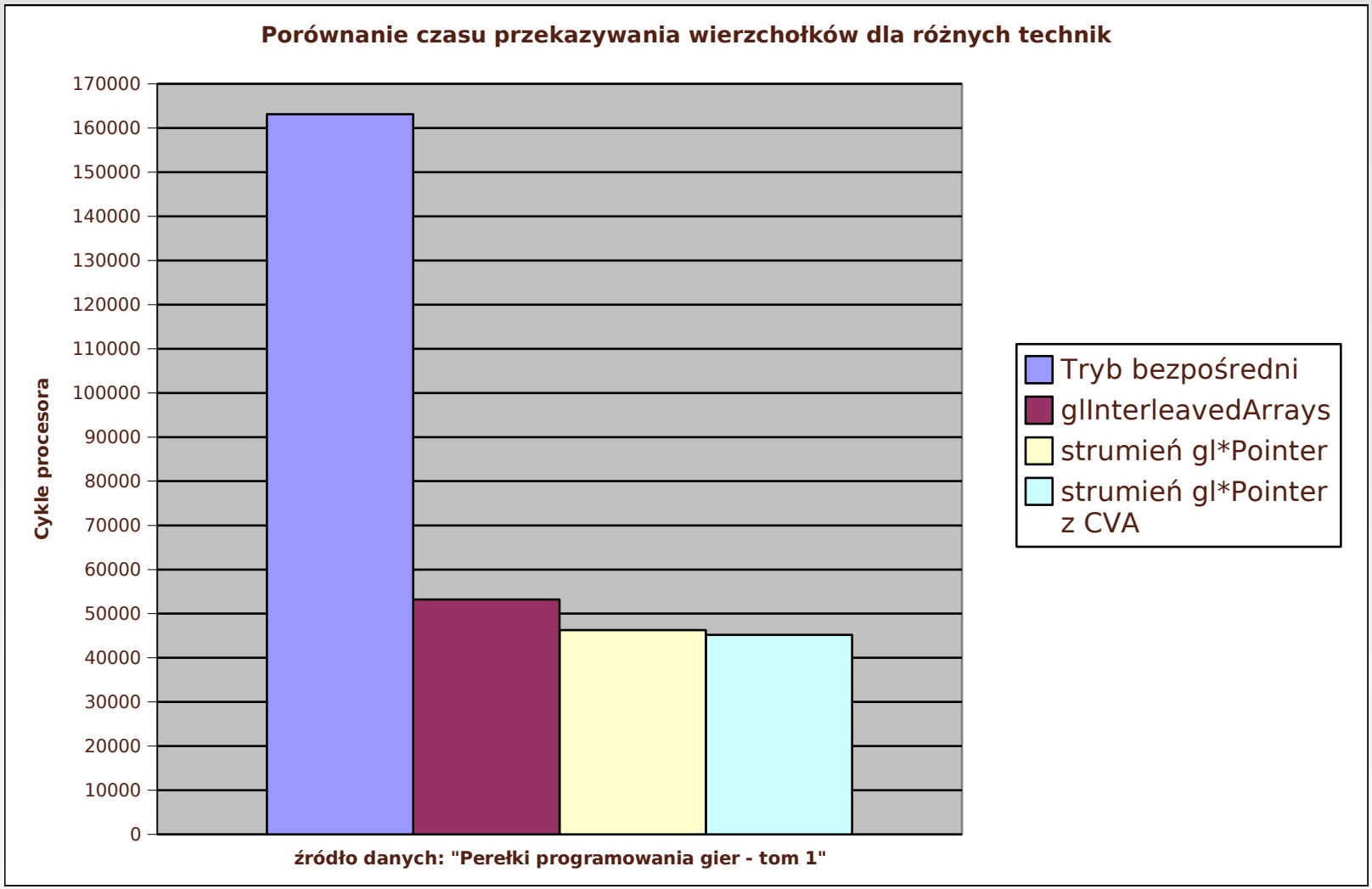
Efektywne renderowanie

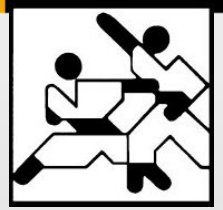
Metody dostarczania wierzchołków do karty graficznej:

- tryb bezpośredni
  - dużo wywołań funkcji = duży narzut czasowy
- dane przeplatane
  - glInterleavedArrays
- dane krokowe i strumieniowe, CVA
  - glVertexPointer, glColorPointer,...
- rozszerzenia producentów, VBO



Efektywne renderowanie

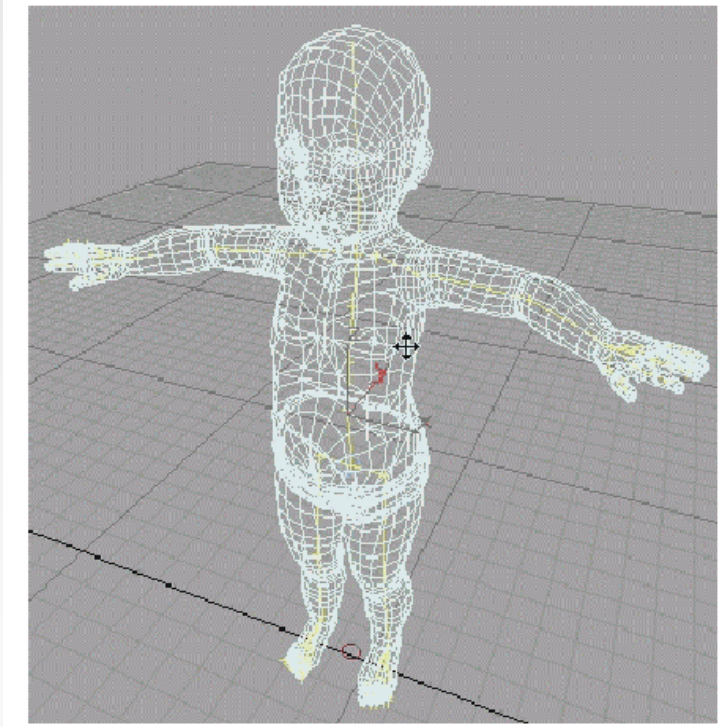


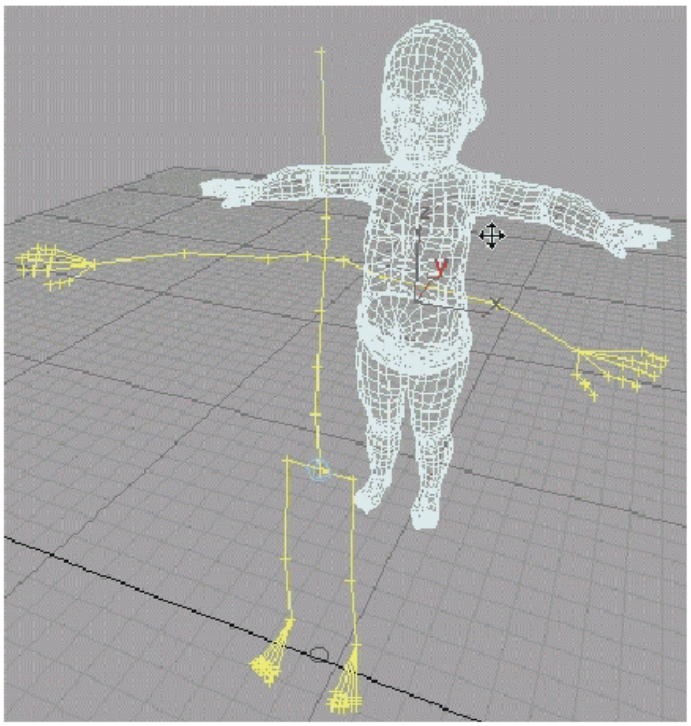


Animacja szkieletowa

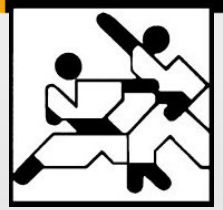
animowany model

- skóra - siatka złożona z wielokątów
- szkielet - pełni funkcję pomocniczą, nie jest renderowany





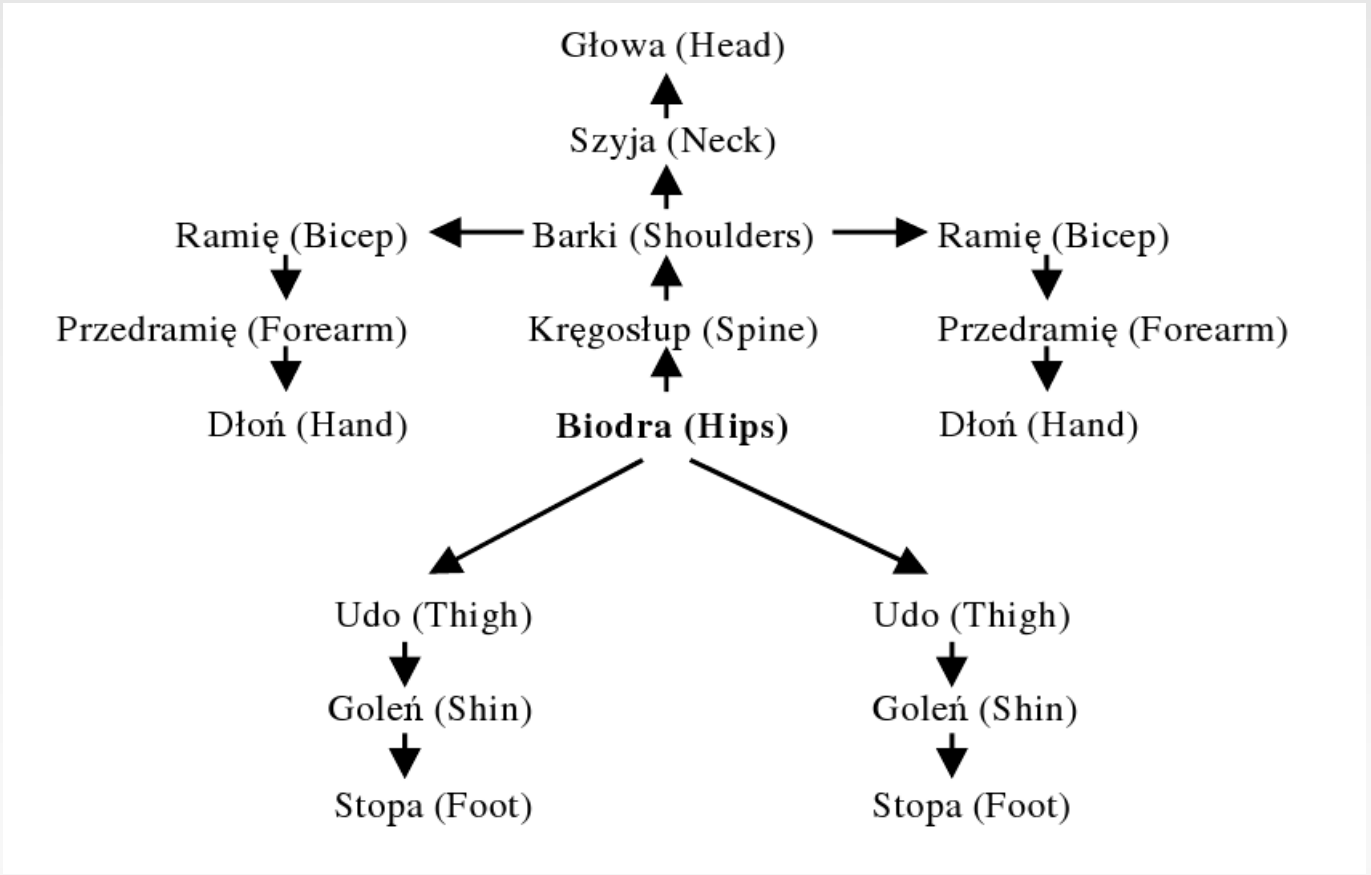
źródło: sound.eti.pg.gda.pl/student/sdio/1 -Animacja.pdf



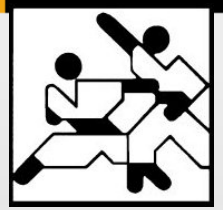
Animacja szkieletowa

hierarchiczna struktura kości

- animowanie sekwencyjne
- odwrotna kinematyka



źródło: sound.eti.pg.gda.pl/student/sdio/1 -Animacja.pdf

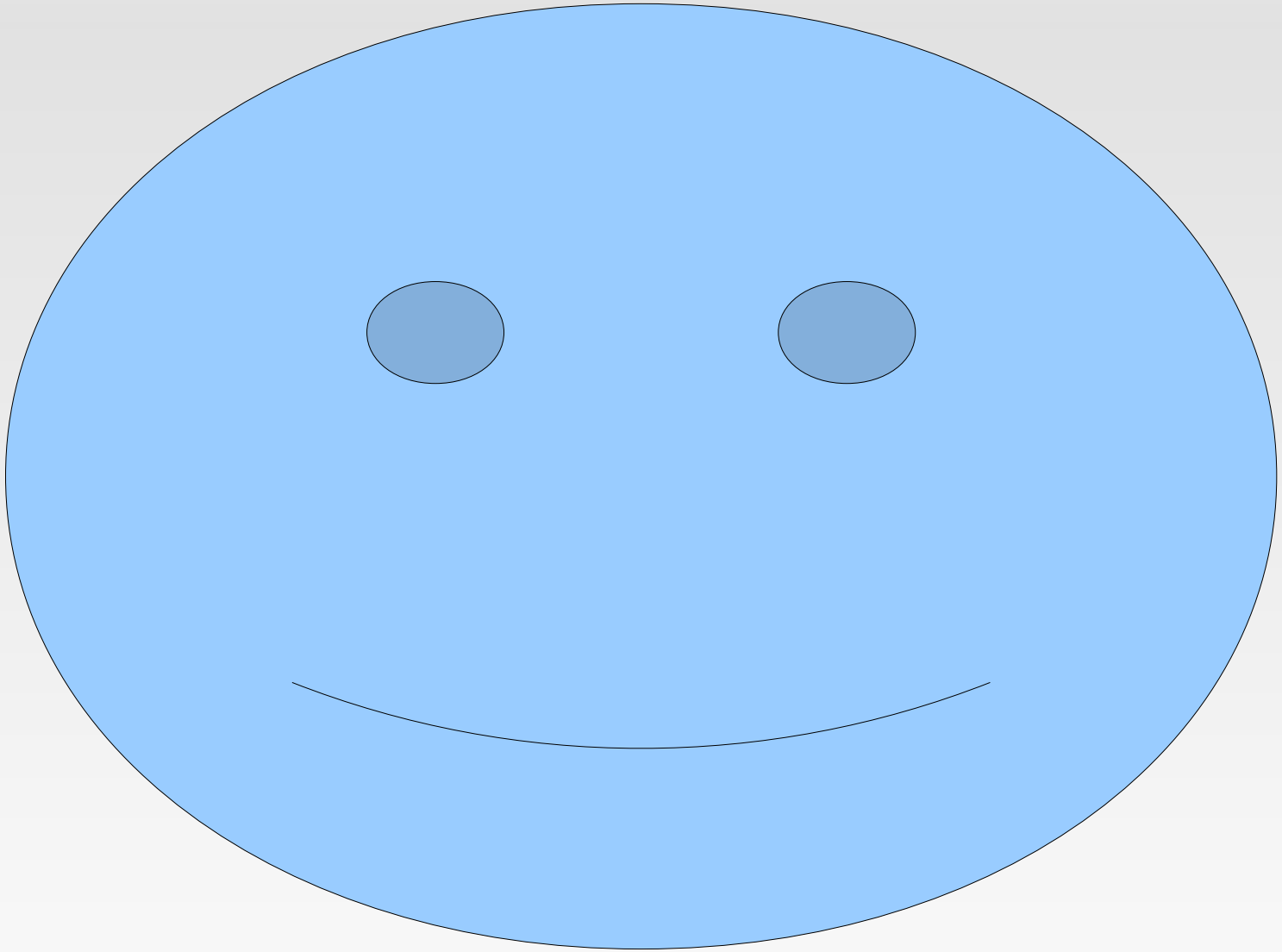


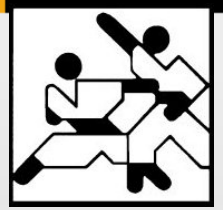
Animacja szkieletowa

algorytm animacji szkieletowej

- aktualizacja szkieletu
- dla każdego wierzchołka, na podstawie ważonej transformacji modyfikujących go kości, obliczana jest pozycja wypadkowa

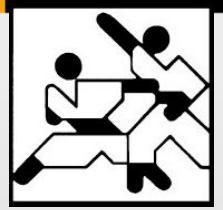
Moja praca magisterska





Literatura

- „Perełki programowania gier – tom 1” pod redakcją Marka DeLoura – Helion 2002
- Animacja szkieletowa z wykorzystaniem shaderów wierzchołków grafika.iinf.polsl.gliwice.pl/doc/09-SKE.pdf
- Podstawy animacji komputerowej sound.eti.pg.gda.pl/student/sdio/12-Animacja.pdf



Game over

Dziękuję za uwagę.
Pytania?