VEKTORIA-MANUAL

DEVICES

Prof. Dr. Tobias Breiner

vektoria

www.vektoria-engine.com

Inhalt der Vorlesung

- VERTIEFUNG
- IO-DEVICES
- TASTATUR
- MAUS & CURSOR
- GAME CONTROLER

VERTIEFUNG

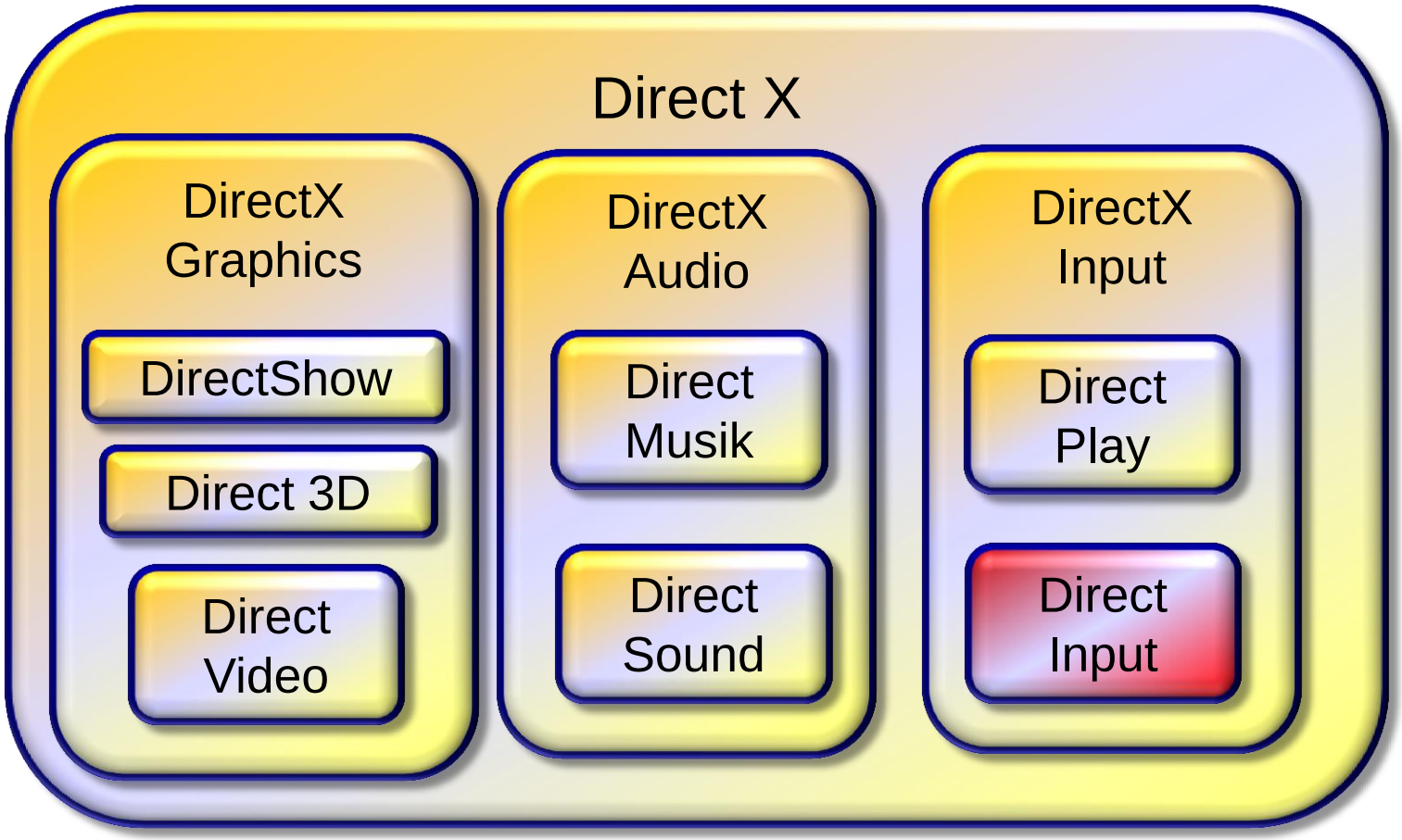
2

3

4

5

Direct Input



Kapitel 2

1

IO-DEVICES

3

4

5

Devices

Devices sind Ein- und Ausgabegeräte

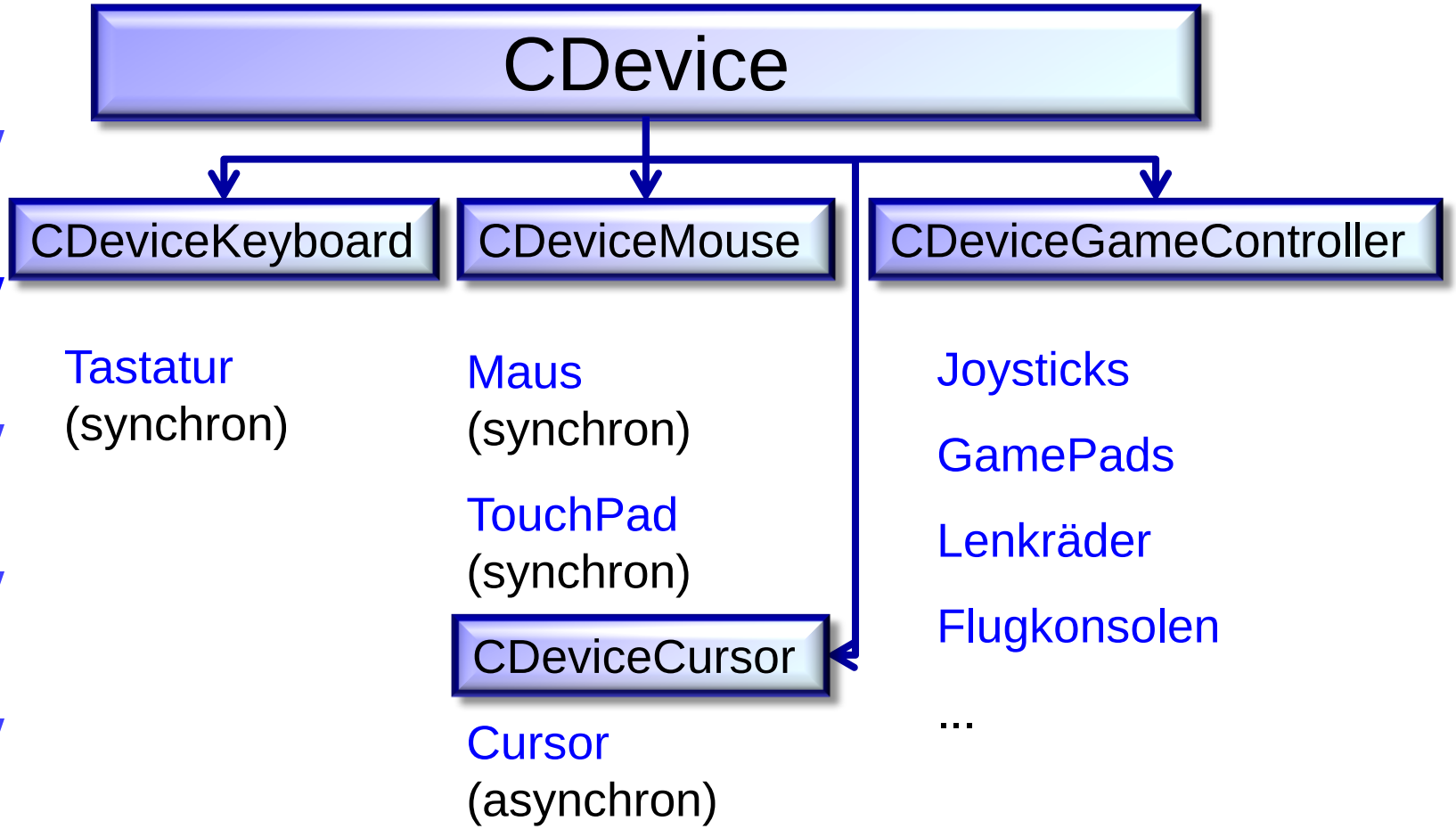
Synonyme in anderen Szenegrafen:

Input Devices, Inputs

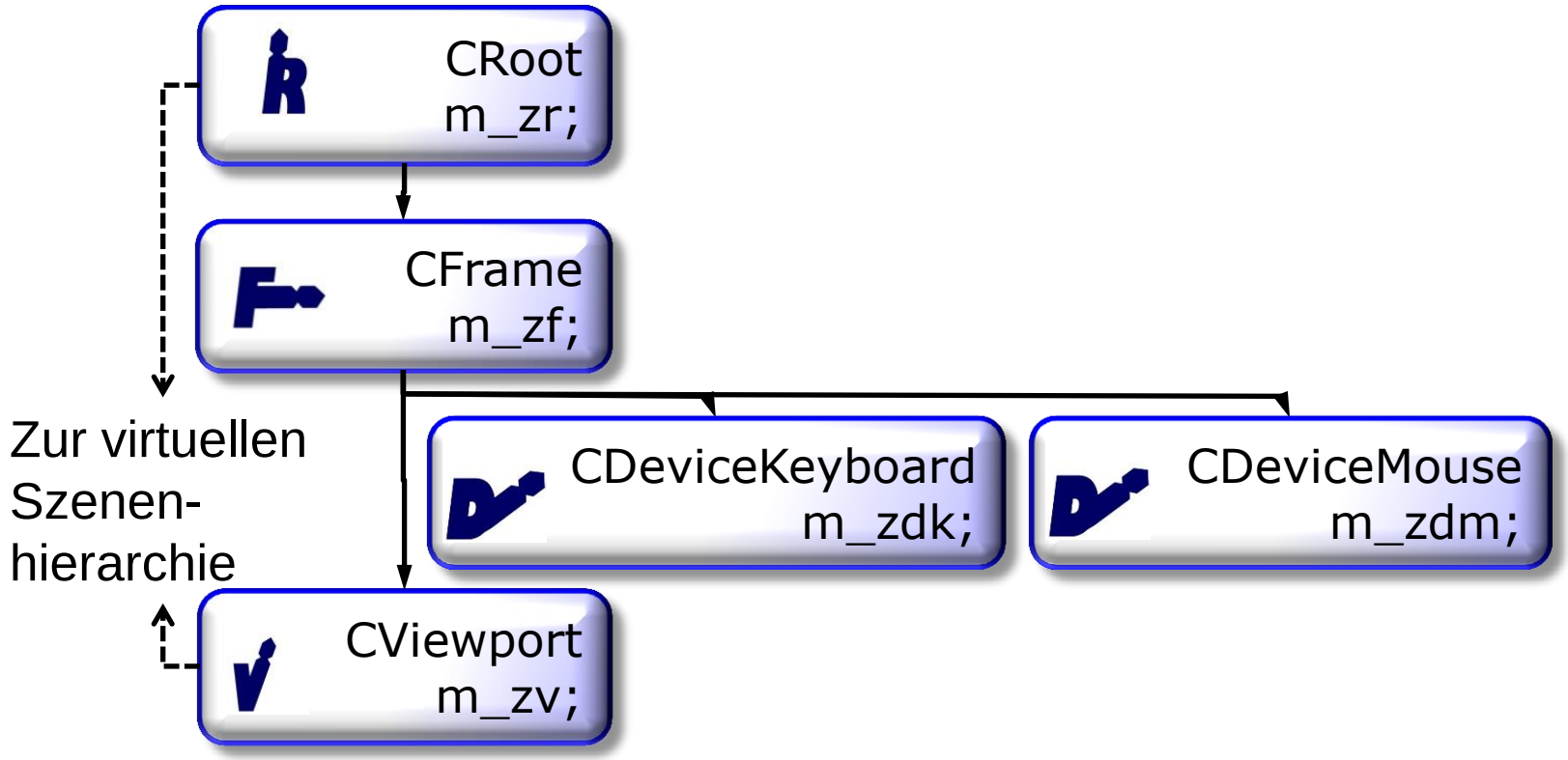
Beispiele:

- Tastatur
- 3D Maus
- Maus
- Cursor (nicht zu verwechseln mit Maus)
- TouchPad
- Lenkrad
- Joystick mit und ohne Force Feedback
- Game Controller
- Motion Base
- …

Abgeleitete Klassen von CDevice



Beispiel für Devicehierarchie



Add-Methoden des Frames für Eingabe

Hat man den Frame für die Eingabe initialisiert, muss noch das jeweilige Device angehangen werden, dafür gibt es folgende vier Methoden in CFrame:

```
void AddDeviceKeyboard
        (CDeviceKeyboard * pdevicekeyboard);
void AddDeviceMouse
        (CDeviceMouse * pdevicemouse);
void AddDeviceGameController
        (CDeviceGameController * pdevicegamecontroller);
void AddDeviceCursor
        (CDeviceCursor * pdevicecursor);
```

Kapitel 3

1

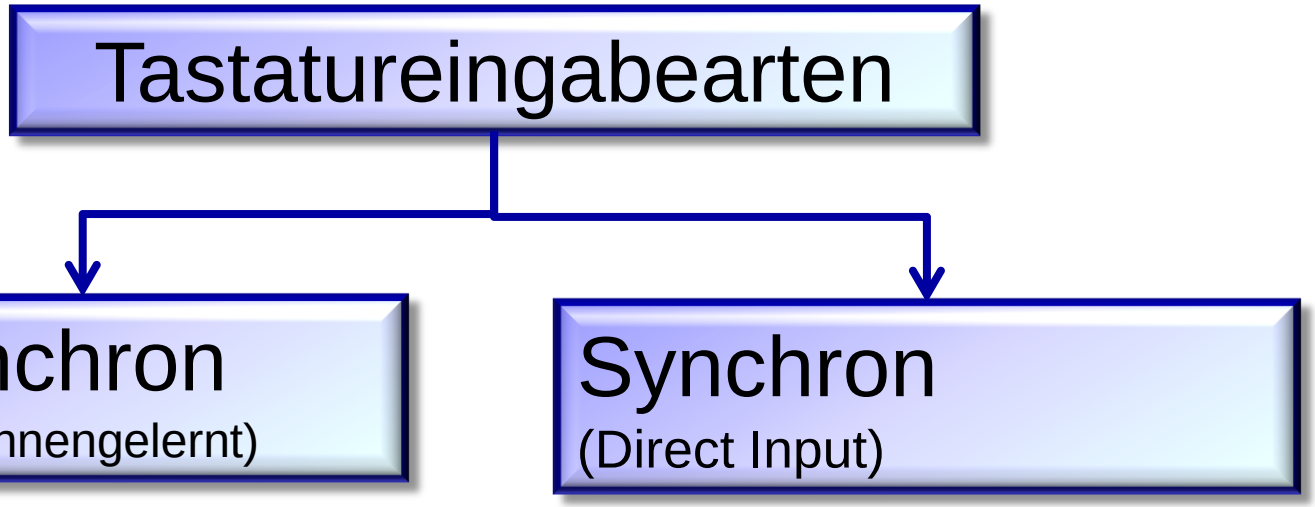
2

TASTATUR

4

5

Tastatureingabearten



Warte, bis eine Taste gedrückt ist, und stelle eine Meldung in die Message-Queue für die Applikation!

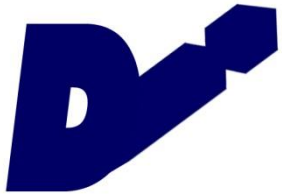
Geeignet für Texteingabe

Meldet der Applikation sofort, wenn eine Taste gedrückt ist.

Geeignet für Games und VR-Interaktion

Tastatursteuerung

CDeviceKeyboard::PlaceWASD



bool PlaceWASD(CPlacement & zp, float & fTimeDelta,
bool bEarth);

1

2

3

4

5

Diese Methode der Klasse CInputX implementierte eine komplette WASD-Steuerung. Sehr praktisch!

fTimeDelta ist diejenige Zeit, die seit dem letzten Frame vergangen ist.

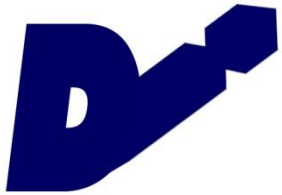
Gibt true aus, wenn Leertaste gedrückt wurde. (z.B. für Shooter)

Placement der Kamera, wird von der Funktion automatisch jedes Frame befüllt

true, wenn Szene auf der Erde spielt (Horizont bleibt dann horizontal), im Weltall false.

Tastatursteuerung

CDeviceKeyboard::PlaceWASD



bool PlaceWASD(CPlacement & zp, float & fTimeDelta,
bool bEarth);

Tasten der WASD-Steuerung:

A:	nach links
W:	nach vorne
S:	nach hinten
D:	nach rechts
R:	nach oben
F:	nach unten
Rechtspfeil:	Rechtsdrehung (yaw right)
Linkspfeil:	Linksdrehung (yaw left)
Aufpfeil:	Hochdrehung (pitch up)
Abpfeil:	Runterdrehung (pitch up)
Leertaste:	Programmabhängige Sonderaktion

CDeviceKeyboard, WASD-Hilfsmethoden

```
void SetWASDTranslationSensitivity (float fTranslationSensitivity);
float GetWASDTranslationSensitivity ();
```

Setzt bzw. holt die Verschiebungs-Empfindlichkeit (Translationsgeschwindigkeit) bei den WASD-Tasten in Einheiten pro Sekunde für die Methode PlaceWASD

```
void SetWASDRotationSensitivity (float fRotationSensitivity);
float GetWASDRotationSensitivity();
```

Setzt bzw. holt die Drehempfindlichkeit (Rotationsschnelligkeit) bei den Pfeil-Tasten in Bogenmaß pro Sekunde für PlaceWASD

CDeviceKeyboard, WASD-Hilfsmethoden

Erzeugt Kameraminimalhöhenlevel für die Erdsteuerung, die Kamera kann nie unter fyLevelMin untertauchen.

```
void SetWASDLevelMin(float fyLevelMin);
void SetWASDLevelMax(float fyLevelMax);
```

Erzeugt Kameramaximalhöhenlevel für die Erdsteuerung, die Kamera kann nie über fyLevelMax steigen.

```
float GetWASDLevelMin();
float GetWASDLevelMax();
```

Und natürlich gibt‘s dazu noch die entsprechenden Hol-Methoden.

CDeviceKeyboard::KeyPressed

bool KeyPressed(int iKey);

true, falls Taste iKey (Tastaturnummer gemäß DirectX) gedrückt wurde, ansonsten false

Beispiel:

```
if(KeyPressed(DIK_K))
{
  // geht hier rein, wenn Taste „K“ gedrückt wurde
}
if(KeyPressed(DIK_K))
{
  // geht hier rein, wenn Taste „K“ gedrückt wurde
}
```

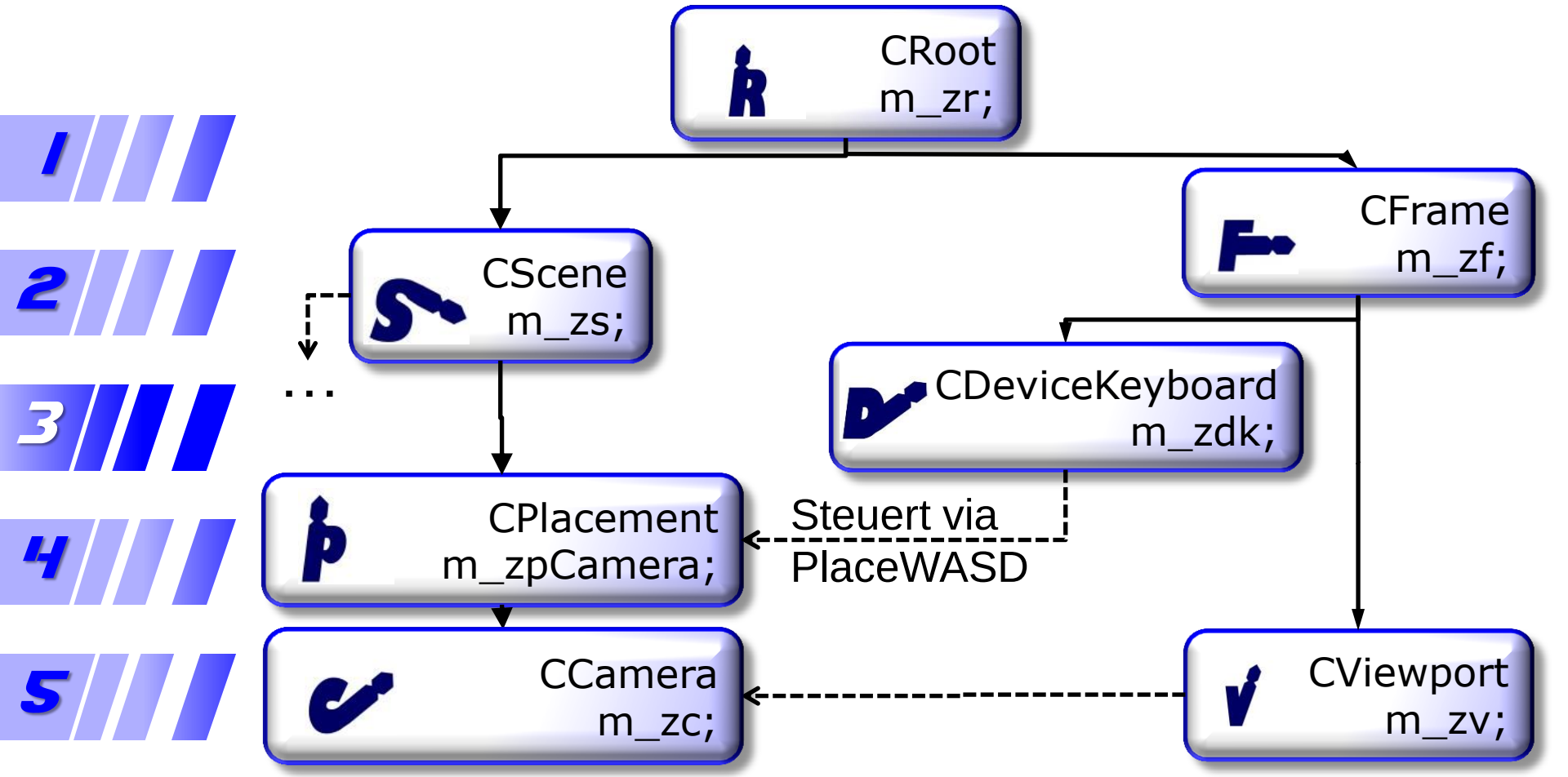
Übung

- Programmieren Sie eine WASD-Steuerung für ihre Erde!

Freiwillige Zusatzaufgabe für die schnellen Nerds:

- Nach Drücken der Leertaste soll sich die Erde gemäß WASD bewegen, nochmaliges Drücken bewegt wieder die Kamera.

Lösung Tastatursteuerung



Lösung Tastatursteuerung (2/5)

Vektoria-Objekte in CGame.h

1

2

3

4

5

```
CRoot m_zr;
CScene m_zs;
CPlacement m_zpCamera;
CPlacement m_zpSphere;
CGeoSphere m_zgSphere;
CFrame m_zf;
CViewport m_zv;
CCamera m_zc;
CParallelLight m_zl;
CMaterial m_zm;
CDeviceKeyboard m_zdk;
```

Lösung Tastatursteuerung (3/5)

Init-Methode in CGame.cpp

```
void CGame::Init(HWND hwnd,
                 HWND hwndDX, RECT rectWnd)
{
  m_zf.Init(hwnd,rectWnd.right,rectWnd.bottom,
            eRenderApi_DirectX11_Shadermodel50,
            eInputApi_DirectInput);
  m_zv.InitFull(&m_zc);
  m_zr.AddFrameHere(&m_zf);
  m_zf.AddViewport(&m_zv);
  m_zf.AddDeviceKeyboard(&m_zdk);
  m_zm.MakeTextureImage
       ("textures\\earth_image.jpg");
  m_zl.Init(CHVector(1,0,1),CColor(1,1,1));
  …
```

1 2 3 4 5

Lösung Tastatursteuerung (4/5)

Init-Methode in CGame.cpp

```
  …
  m_zr.AddScene(&m_zs);
  m_zs.AddPlacement(&m_zpCamera);
  m_zs.AddParallelLight(&m_zl);
  m_zs.AddPlacement(&m_zpSphere);
  m_zpCamera.AddCamera(&m_zc);
  m_zpSphere.Translate(CHVector(0,0,-3));
  m_zpSphere.AddGeo(&m_zgSphere);
  m_zgSphere.Init(2.0F,&m_zm);
}
```

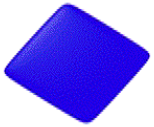
Lösung Tastatursteuerung (5/5)

Tick-Methode in CGame.cpp

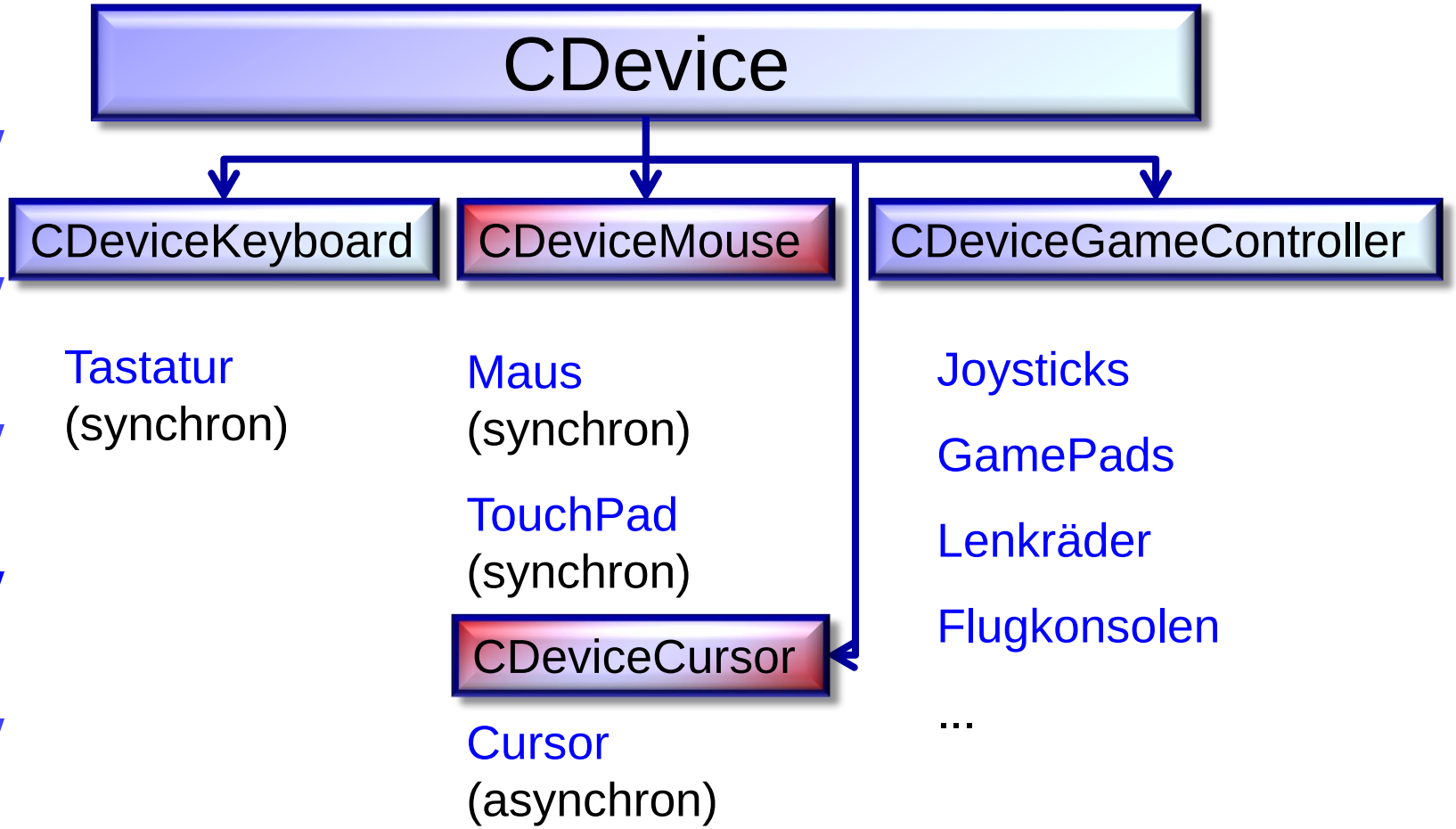
```
void CGame::Tick(float fTimeDelta)
{
  m_zdk.PlaceWASD(m_zpCamera,fTimeDelta);
  m_zr.Tick(fTimeDelta);
}
```

Kapitel 4

1
2
3
MAUS & CURSOR
5



Maus und Cursor



Unterschied zw. Maus und Cursor

Achtung! Maus und Cursor werden häufig miteinander verwechselt.

Während eine Maus jedoch ein reales Eingabegerät ist, stellt ein Cursor eine zeichnerische Interpretation der Mausbewegungen dar.

In Vektoria wird dieser Umstand durch zwei verschiedene Device-Klassen Rechnung getragen: CDeviceMouse und CDeviceCursor

Wenn Sie absolute Cursorkoordinaten bezüglich des Fensters haben wollen, dann verwenden Sie CDeviceCursor
Wenn es auf schnelle relative Mausbewegungen ankommt, verwenden Sie CDeviceMouse

Mausbewegungslesemethoden

```
float GetRelativeX(void);
// Gibt relative Mausbewegung in X-Richtung aus
float GetRelativeY(void);
// Gibt relative Mausbewegung in Y-Richtung aus
float GetRelativeZ(void);
// Gibt relative Mausradbewegung (Z-Richtung) aus

float GetAbsoluteX(void);
// Gibt die absolute Mausbewegung auf der X-Achse
aus (nicht zu verwechseln mit Cursorpos!)
float GetAbsoluteY(void);
// Gibt die absolute Mausbewegung auf der y-Achse
aus (nicht zu verwechseln mit Cursorpos!)
```

Synchr. Mausklicklesemethoden

```
bool ButtonPressed(int a_iButton);
// true, falls Taste iButton (Tastennummer gemäß
DirectX) gedrückt wurde, ansonsten false
(synchron)
```

Parametrisierungsmethoden

```
void SetSensitivity(float fSensitivity=0.0005f);
// Setzt die Empfindlichkeit

float GetSensitivity(void);
// Gibt die Empfindlichkeit aus

void SetInvertedYAxisOn();
// Y-Achse invertieren

void SetInvertedYAxisOff();
// Y-Achse nicht invertieren
```

Übung Mauseingabe

- Lassen Sie die Erdkugel zusätzlich durch die Maus drehen!

Freiwillige Zusatzaufgabe für die schnellen Nerds:

- Implementieren Sie eine progressive Maussteuerbewegung, d.h. erst bewegt sich die Erde langsam, hat sie aber einen gewissen Drall, dann immer schneller!

Cursorpositionslesemethoden

```
bool GetAbsolute(int & ix, int & iy,
       bool bHideCursor = false);
// Gibt die Zeigerpositionskoordinaten bezüglich
// der linken oberen Ecke des Frames aus, gibt
// true aus, wenn Cursor sich innerhalb des Frames
// befindet
bool GetFractional(float & frx, float & fry,
       bool bHideCursor = false);
// Gibt die fraktionalen Zeigerpositions-
// koordinaten (Wertebereich jeweils 0..1)
// bezüglich des Frames aus, gibt true aus, wenn
// Cursor sich innerhalb des Frames befindet.
// Ist HideCursor true, dann verschwindet er, wenn
// er sich im Frame befindet => Man kann mit
// Overlays seinen eigenen Cursor erzeugen
```

Assynchr. Mausklicklesemethoden

```
bool ButtonPressed(int iButton);
// true, falls Taste iButton gedrückt wurde,
ansonsten false

bool ButtonPressedLeft(); // true, falls linke
Maustaste gedrückt wird, ansonsten false
bool ButtonPressedRight(); // true, falls rechte
Maustaste gedrückt wird, ansonsten false
bool ButtonPressedMid(); // true, falls mittlere
Maustaste gedrückt wird, ansonsten false
```

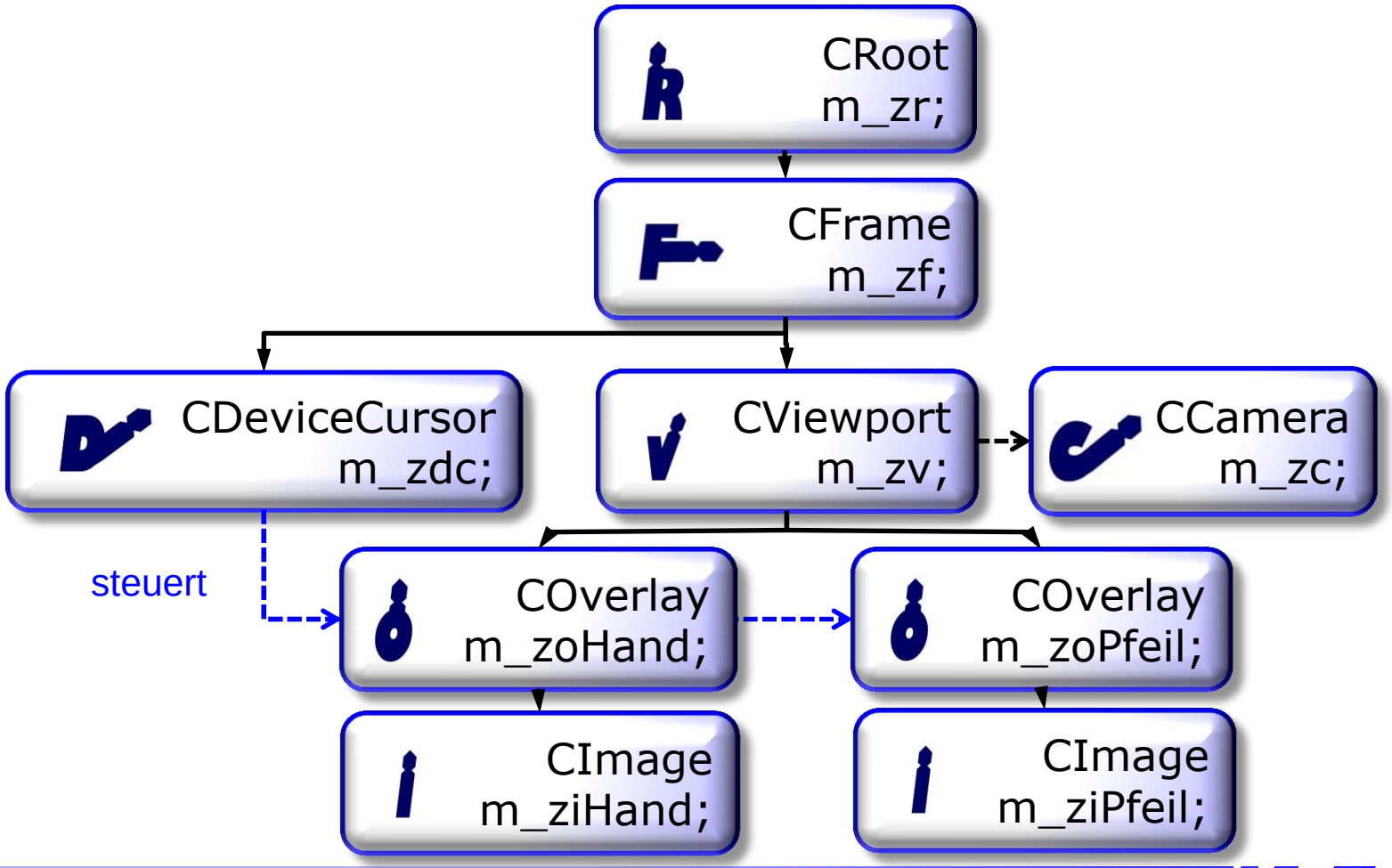
Übung Cursor ersetzen

- Ersezten Sie den Windows-Cursor durch Ihren eigenen!

Freiwillige Zusatzaufgabe für die schnellen Nerds:

- Wenn die linke Maustaste gedrückt ist, soll eine Hand erscheinen ansonsten ein Pfeil!

Lösung Cursor Ersetzen



Nerd-Lösung Cursor Ersetzen (2/5)

Vektoria-Objekte in CGame.h

1

2

3

4

5

```
CRoot m_zr;
CFrame m_zf;
CDeviceCursor m_zdc;
CViewport m_zv;
CCamera m_zc;
CImage m_ziPfeil;
CImage m_ziHand;
COverlay m_zoPfeil;
COverlay m_zoHand;
```

Nerd-Lösung Cursor Ersetzen (3/5)

Init-Methode in CGame.cpp

```
m_zf.Init(hwnd, rectWnd.right,
                rectWnd.bottom);
m_ziPfeil.Init("Pfeil.bmp");
m_ziHand.Init(„Hand.bmp");

m_zoPfeil.Init(&m_ziPfeil,
      CFloatRect(0.0F,0.0F,0.02F,0.02F),true);
m_zoHand.Init(&m_ziHand,
      CFloatRect(0.0F,0.0F,0.02F,0.02F),true);
m_zv.InitFull(&m_zc);
m_zr.AddFrameHere(&m_zf);
m_zf.AddViewport(&m_zv);
m_zf.AddDeviceCursor(&m_zdc);
m_zv.AddOverlay(&m_zoPfeil);
m_zv.AddOverlay(&m_zoHand);
```

Diese Bilder müssen natürlich noch mit Gimp oder Photoshop erzeugt und in den Projekt-ordner gelegt werden.

Nerd-Lösung Cursor Ersetzen (4/5)

Tick-Methode in CGame.cpp

```
float fx, fy;
m_zdc.GetFractional(fx,fy,true);
if(m_zdc.ButtonPressedLeft())
{
    m_zoPfeil.SwitchOff(); m_zoHand.SwitchOn();
    m_zoHand.SetRect(
      CFloatRect(fx-0.01F,fy-0.01F,0.02F,0.02F));
}
else
{
    m_zoPfeil.SwitchOn();m_zoHand.SwitchOff();
    m_zoPfeil.SetRect(
      CFloatRect(fx-0.01F,fy-0.01F,0.02F,0.02F));
}
m_zr.Tick(fTimeDelta);
```

ReSize-Methode in CGame.cpp


```
void CGame::WindowReSize
                (int iNewWidth, int iNewHeight)
{
        m_zf.ReSize(iNewWidth, iNewHeight);
}
```

Wichtig!
Man muss dem Frame in WindowReSize die Fenstergröße übermitteln, sonst wird der neue Cursor nicht angezeigt!
Ursache: Der Frame gibt die Fenstergröße an den angehängten DeviceCursor weiter dieser berrechnet daraus die Cursorposition, die er zum Anzeigen braucht.

Picking allgemein

1. Picking bezeichnet das Auswählen von Objekten mit Hilfe des Cursors – in diesem Falle das Auswählen von 3D-Objekten.
2. Dazu geht man mit der Maus über ein Objekt und drückt auf eine Maustaste – normalerweise die linke Maustaste.
3. Picking wird nur von wenigen Szenegrafen und Engines unterstützt.
4.
5.

Picken des Viewports in Vektoria

In Vektoria findet man die Picking-Methoden in der Klasse CDeviceCursor. Zum Picken des Viewports gibt es folgende Methoden:

1
2
3
4
5

```
CViewport * PickViewport();
// Gibt erstes sichtbares Viewport aus,  welches
// unter dem Cursor liegt, falls kein Viewport
// gefunden wurde, ist das Ergebnis NULL.

CViewport * PickViewport(  float & frxViewport,
                           float & fryViewport);
// Gibt erstes sichtbares Viewport aus,  welches
// unter dem Cursor liegt, falls kein Viewport
// gefunden wurde, ist das Ergebnis NULL.
// Zusäztlich werden die passenden
// Bildschirmkoordinaten frxViewport und fryViewport
// des Picking-Punktes mit ausgegeben.
```

Picken von Overlays und Kameras

Zum Picken eines Overlays gibt es folgende Methode:

```
COverlay * PickOverlay();
// Gibt oberstes sichtbares Overlay aus,
// welches unter dem Cursor liegt, falls kein
// Overlay gefunden wurde ist das Ergebnis NULL.
```

Und zum Picken einer Kamera folgende Methode:

```
CCamera * PickCamera();
// Gibt die Kamera aus, die mit dem sichtbaren
// Viewport verbunden ist, welches unter dem Cursor
// liegt, falls kein Viewport gefunden wurde ist das
// Ergebnis NULL.
```

Picken einer Szene

Zum Picken einer Szene gibt es folgende Methode:

```
CScene * PickScene();
// Gibt die Szene aus, welche das sichtbare Viewport
// zeigt, welches unter dem Cursor liegt, falls
// keine Szene gefunden wurde ist das Ergebnis NULL.
```

Picken von Placements

Zum Picken eines oder mehrerer Placements gibt es folgende Methoden:

1.
2.
3.
4.
5.

```
CPlacement * PickPlacement();
// Gibt dasjenige näheste Placement mit der
// niedersten Hierarchiestufe aus, welches unter dem
// Cursor liegt. Falls kein Placment gefunden wurde,
// ist das Ergebnis NULL.

PickPlacements(CPlacements * zps);
// Gibt alle Placements aus, welche unter dem
// Cursor liegen und schreibt sie in den Container
// zps.
```

Die Placements werden anhand ihrer jeweiligen Bounding-Box gefunden.

Picken von Geometrien

Zum Picken einer Geometrie gibt es folgende Methode:

```
CGeo * PickGeo(CHVector & vIntersection,
               float & fDistanceSquare);
// Gibt dasjenige Geo mit der niedersten Hierarchie-
// stufe aus, welches unter dem Cursor liegt. Falls
// kein Geo gefunden wurde, ist das Ergebnis NULL.
// vIntersection ist der genaue Schnittpunkt,
// fDistanceSquare ist das Quadrat der Entfernung
// zwischen Strahlursprung und Schnittpunkt.
```

Diese Methode dauert lange. Sie kann bei sehr großen Geometrien durchaus einige Millisekunden dauern. Sie sollte daher nur dann angewandt werden, wenn exaktes Picking zwingend notwendig ist, ansonsten sollte einfach das darüberliegende Placement gepickt werden.

Picken mit Schnittpunktdaten

Wenn beim Picken einer Geometrie auch noch weitere Daten zum genauen Schnittpunkt benötigt werden eignet sich folgende Methode:

```
CHitPoint * PickHitPoint();
```

Die Klasse CHitPoint enthält folgende Zusatzdaten:

- Position des Schittpunktes zwischen Pickstrahl und Geometrie.
- UV-Koordinaten des Schnittpunktes.
- Gemitteltre Oberflächennormale.
- Distanz zwischen Kamera und Schnittpunkt.

Picken (Pluralklassen)

Zu vielen Picking-Methoden, die nur das erste Objekt picken, welches der Pickstrahl trifft, gibt es auch nich die entsprechenden Pluralmethoden, welche alle ensprechenden Objekte ermittelt, die der Pickstrahl trifft:

```
void PickOverlays(COverlays * pzos);

void PickPlacements(CPlacements * pzps,
    bool bPickOnlyPlacementsWithDirectGeos = false);

void PickGeos(CGeos * pzgs,
    int iMaxVertices = INT_MAX);

void PickHitPoints(CHitPoints * phitpoints,
    int iMaxVertices = INT_MAX);
```

Picken (Preselected)

Will man nur Objekte aus einer Vorauswahl picken, kann man auf die entsprechenden Preselected-Routinen zurückgreifen:

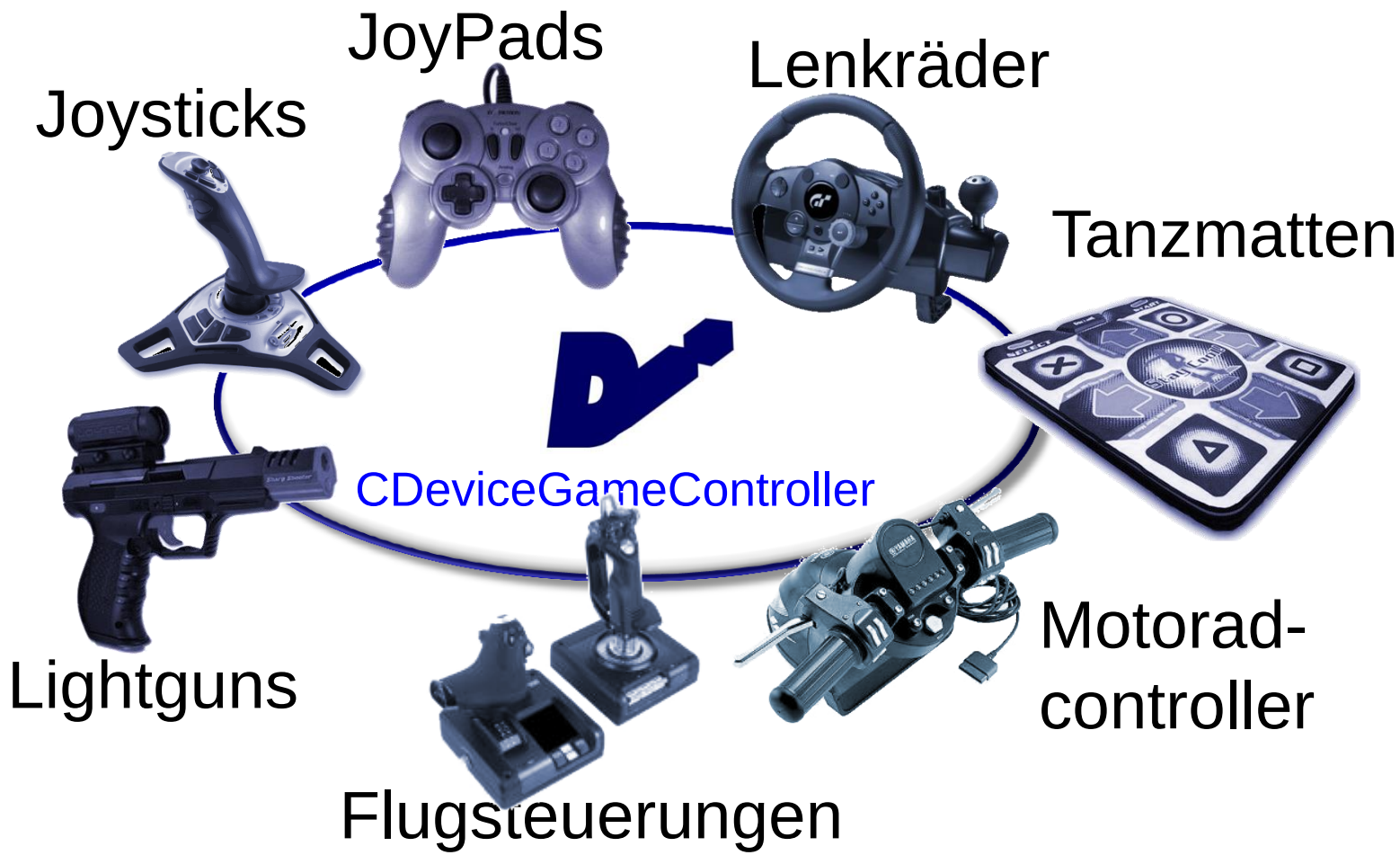
1. COverlay * PickOverlayPreselected(COverlays & zos);
2. CPlacement * PickPlacementPreselected(CPlacements & zps);
3. CGeo * PickGeoPreselected(CGeos & zgs);
4. CHitPoint * PickHitPointPreselected(CGeos & zgs);
5. 

Die Preselected-Routinen eignen sich auch zur Beschleunigung des Pickings.
Insbesondere PickGeoPreselected und PickHitPointPreselected eignen sich für genaues und gleichzeitig performantes Picking.

Kapitel 5

1
2
3
4

Game Controller



Lesemethoden

```
float GetRelativeX(void);
// Gibt relative Controllerbewegung
// in X-Richtung aus
float GetRelativeY(void);
// Gibt relative Controllerbewegung
// in Y-Richtung aus
float GetRelativeZ(void);
// Gibt relative Controllerbewegung
// in Z-Richtung aus

bool ButtonPressed(int a_iButton);
// true, falls Taste iButton (Tastennummer gemäß
DirectX) gedrückt wurde, ansonsten false
```

Parametrisierungsmethoden

```
void SetSensitivity(float fSensitivity=0.0005f);
// Setzt die Empfindlichkeit

float GetSensitivity(void);
// Gibt die Empfindlichkeit aus

void SetInvertedYAxisOn();
// Y-Achse invertieren

void SetInvertedYAxisOff();
// Y-Achse nicht invertieren
```

1
2
3
4
5

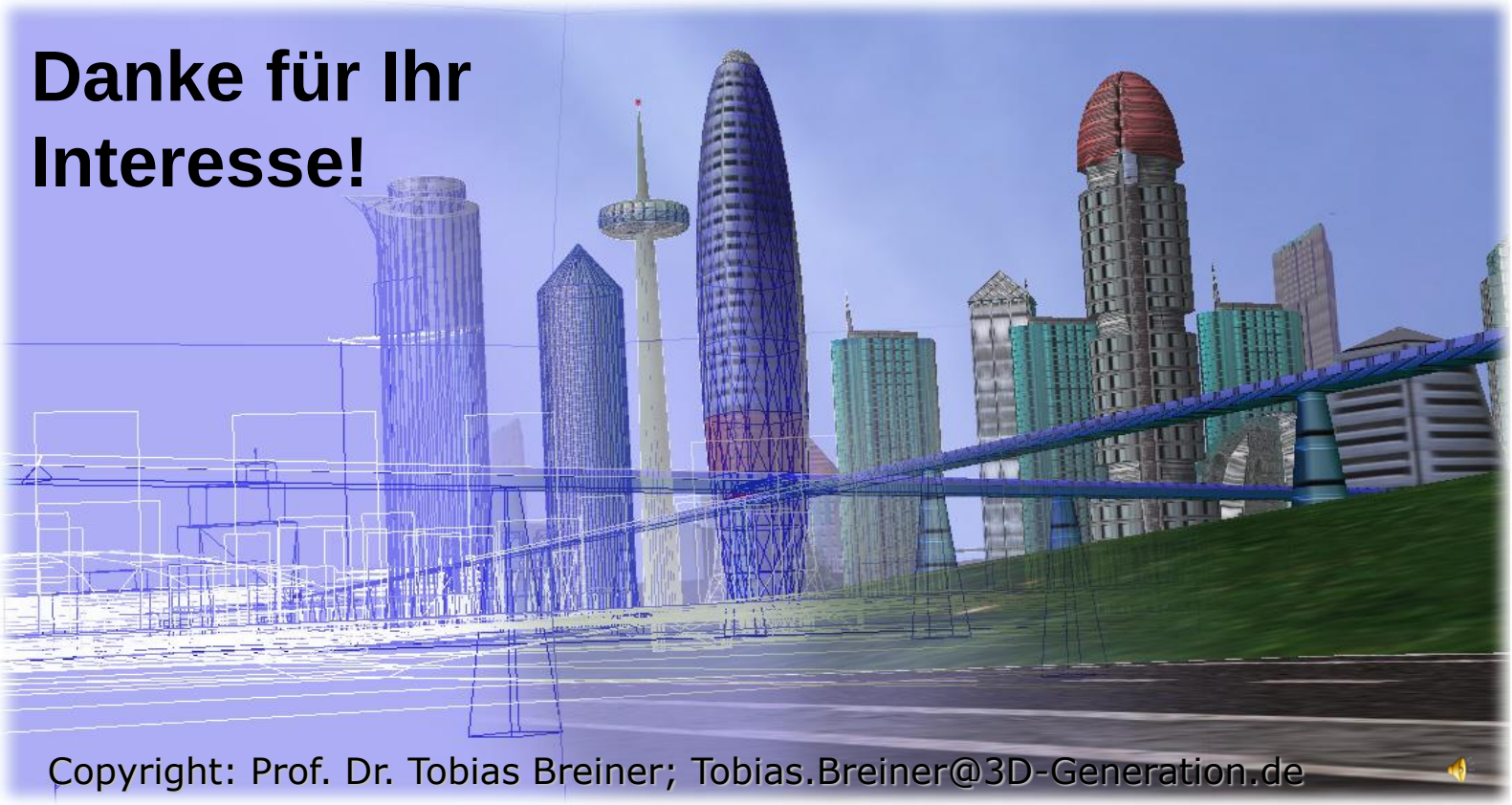
Übung Raumflugsteuerung

- Programmieren Sie eine intuitive Raumflugsteuerung für ein Game Controller Ihrer Wahl!

Freiwillige Zusatzaufgabe für die schnellen Nerds:

- Fügen Sie der Flugsteuerung Beschleunigung und Abbremsung hinzu!

GAME OVER


Danke für Ihr Interesse!
Copyright: Prof. Dr. Tobias Breiner; Tobias.Breiner@3D-Generation.de