

**12. TEXTUREN | DREHBUCH**

**COMPUTERGRAFIK.ONLINE**

Hochschule Furtwangen University | Fakultät Digitale Medien

Betreuer: Prof. Jirka Dell’Oro-Friedl | Projektstudium SoSe 18

Version: 1.4 | Letzte Änderung: 09.12.2018

Autor: Lisa Würstle MKB 5

**Inhaltsverzeichnis**

12.1 (A) Einleitung 2, 3

12.2 (I) Einleitung 4, 5

12.3 (A) Texturkoordinaten 6, 7

12.4 (A) UV-Mapping 8, 9

12.5 (I) UV-Mapping 10

12.6 (A) Mip-Mapping 11, 12

12.7 (A) Bump-Mapping 13

12.8 (A) Normal-Mapping 14

12.9 (A) Displacement-Mapping 15

12.10 (I) Vergleich 16, 17

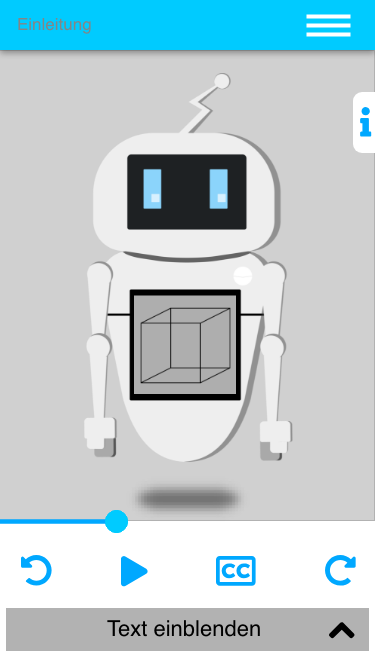
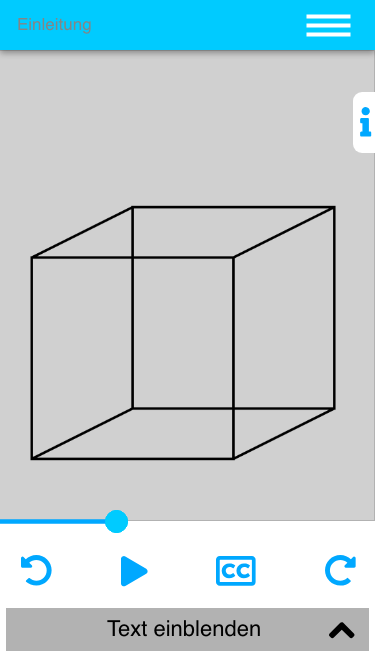
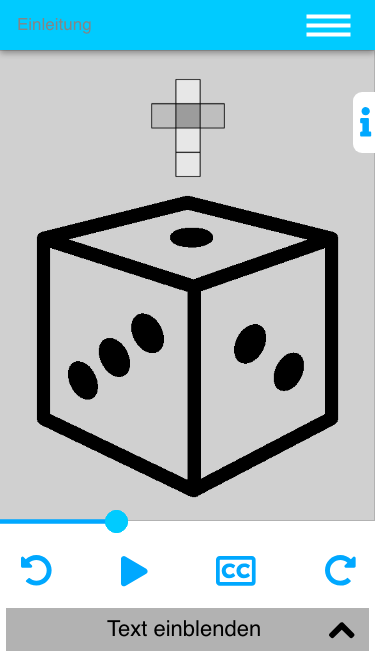
12.11 (A) Environment-Mapping 18

12.12 (A) Kubisches Environment-Mapping 19, 20

**12.1 (A) Einleitung**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 120101  Eine Textur dient im Bereich der Computergrafik dazu Oberflächen von 3D-Objekten realistischer und detailreicher darzustellen.  120102  Die Textur besteht meistens aus einem zweidimensionalen Bild, welches dem 3D-Objekt eine Struktur verleiht. Wie die Textur auf das jeweilige 3D-Objekt projiziert wird, kann auf verschiedene Arten und Weisen geschehen. Diese Verfahren nennt man Mapping-Verfahren.  120103  Der Begriff Mapping bedeutet Zuordnung oder Abbildung. | * Ziel: Oberfläche eines Objekts realistischer wirken zu lassen * Lösung: 2D-Bild = 2D-Textur * verschiedene Mapping-Verfahren * Mapping = Zuordnung / Abbildung | 120101  Einblenden des Roboters  Monitor: Einblenden eines Kubus ohne Texturierung (Fade in Monitor)  Der Kubus fängt an sich um die eigene Achse zu drehen  120102  Eine Spielwürfeltextur erscheint auf dem Kubus (darüber erscheint die aufgeklappte Textur) |

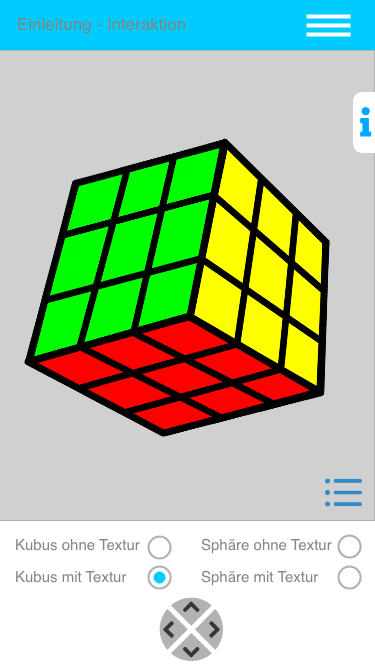
Grafikablauf:

**12.2 (I) Einleitung**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 120201  Wähle den Kubus oder die Sphäre und anschließend eine Textur aus der vordefinierten Liste aus und lasse dir den Vergleich von dem 3D-Objekt ohne und mit Textur anzeigen. Betrachte das 3D-Objekt dabei von allen Seiten. | Wähle den Kubus oder die Sphäre und anschließend eine Textur aus der vordefinierten Liste aus und lasse dir den Vergleich von dem 3D-Objekt ohne und mit Textur anzeigen. Betrachte das 3D-Objekt dabei von allen Seiten. | 120201  Aufgabe wird gesprochen (über das i kann sich der Nutzer die Aufgabe anzeigen lassen)  Klickt der Nutzer auf das Listen-Symbol öffnet sich eine vordefinierte Liste in dem er sich eine Textur aussuchen kann  Der Nutzer kann zwischen den Auswahlmöglichkeiten “Kubus ohne Textur”, “Kubus mit Textur, “Sphäre ohne Textur und “Sphäre mit Textur” auswählen, wobei die Punkte “mit Textur” nur ausgewählt werden können, wenn der Nutzer eine Textur ausgewählt hat  3D-Objekt kann vom Nutzer beliebig über das Steuerkreuz gedreht und von allen Seiten betrachtet werden |

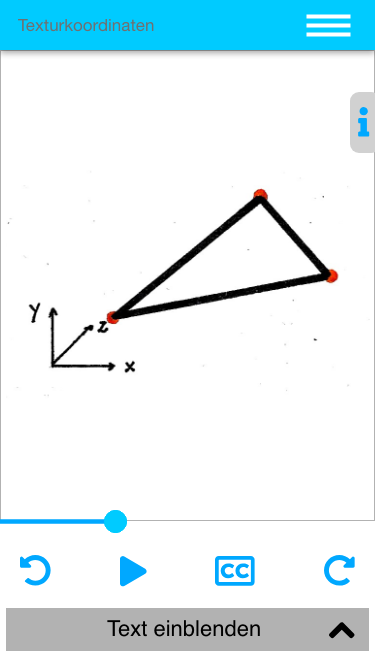
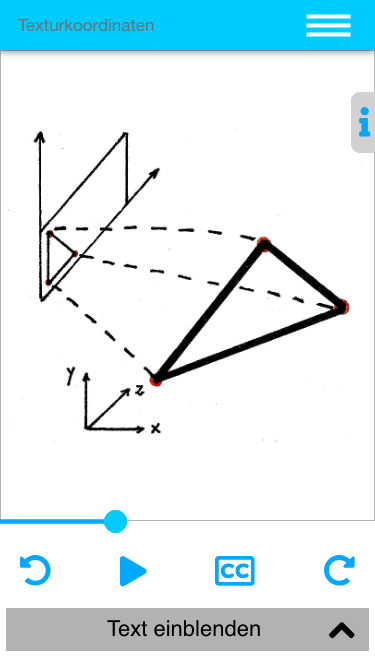
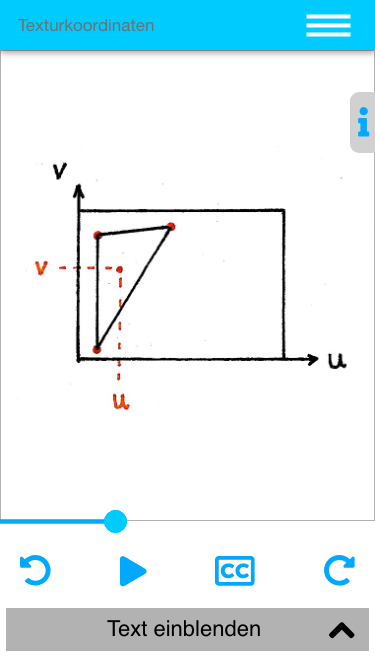
Grafikablauf:

** **

**12.3 (A) Texturkoordinaten**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 120301  Nun stellt sich die Frage, wie die Textur auf das Mesh projiziert wird. Zuerst wird die zu texturierende Fläche im 3D-Raum definiert.  120302  Anschließend ordnet der Benutzer die definierten 3D-Koordinaten mit Hilfe geeigneter Werkzeuge den Vertices des Polygons zu. Die Textur besteht aus Pixeln, welche auch Texel genannt werden.  120303  Texturkoordinaten werden in einem kartesischen Koordinatensystem dargestellt. Die Achsen werden in der Regel mit u und v beschriftet um sie von x und y unterscheiden zu können. Der Wertebereich ist dabei jeweils von 0 bis 1. | * Umwandlung von Objektraum-Koordinaten (x, y, z) in Texturkoordinaten (u, v) * Zuordnung der Koordinaten mit Hilfe geeigneter Werkzeuge * Pixel auf der Textur = Texel | 120301  Einblenden eines Dreiecks im 3D-Raum  Vertices werden markiert  120302  Einblenden eines kartesischen Koordinatensystems in dem eine 2D-Textur liegt  Dreieck wird in 2D-Raum projiziert (Zoom zum Koordinatensystem) und jeder Vertex erhält eine genaue Position auf der Textur  120303  Die Achsen werden mit u und v beschriftet  Ein sichtbarer Pixel erhält eine genaue Texturposition |

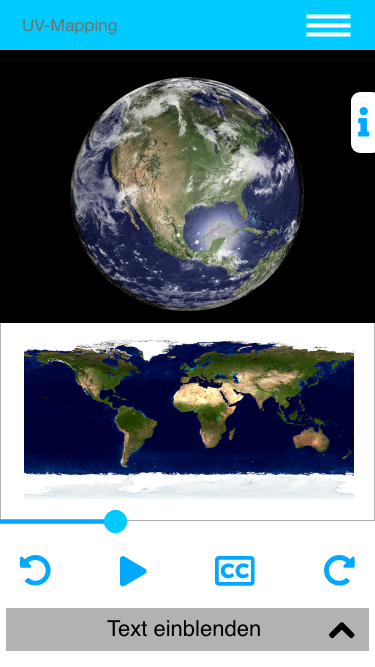
Grafikablauf:

**12.4 (A) UV-Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 120401  Beim UV-Mapping bildet man die Textur mit einer einfachen Abwicklung in der geeigneten Software von Hand. Der Prozess wird auch als Unwrapping bezeichnet.  120402  Bildlich kann man sich die Abwicklung so vorstellen, dass das 3D-Objekt an bestimmten Stellen “aufgeschnitten” und anschließend “abgewickelt” wird, wie in diesem Fall die Weltkugel.  120403  Versuche den Prozess der Abwicklung nachzuvollziehen. | * Texture entsteht durch einfache Abwicklung von Hand * Abwicklung = Unwrapping * Objekt wird “aufgeschnitten” und anschließend “abgewickelt” | 120401  Einblenden der Weltkugel  120402  Weltkugel wird aufgeschnitten und anschließend abgewickelt  Texture Map (siehe Screen) entsteht |

Grafikablauf:

 ****

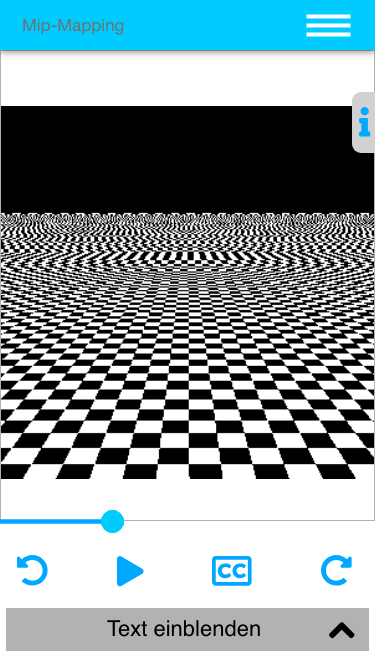
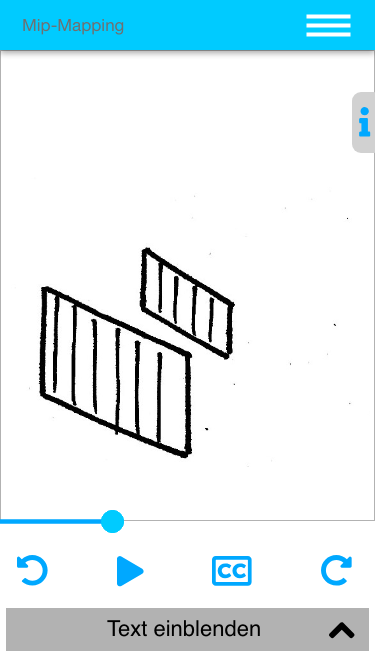
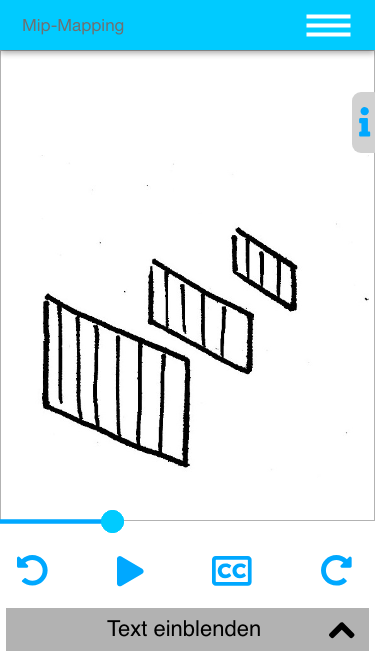
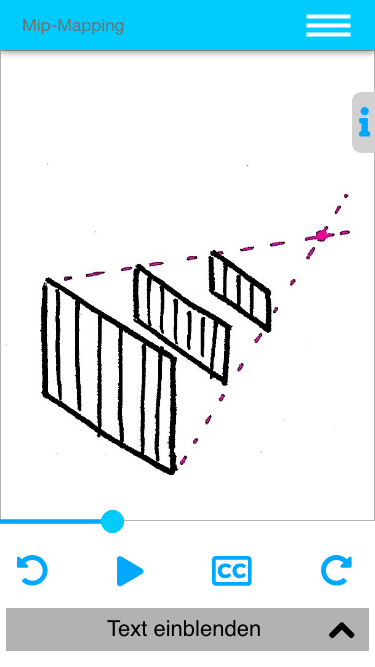
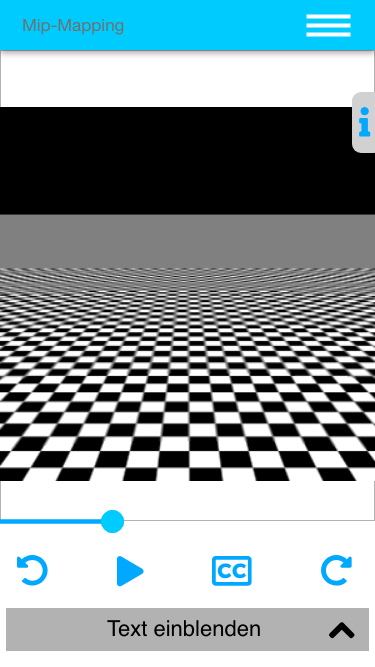
**12.5 (I) UV-Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 120501  Wähle den Pinsel und suche dir eine beliebige Farbe aus. Nun kannst du die von Hand abgewickelte Textur des Roboters oder den Roboter direkt bemalen. | Wähle den Pinsel und suche dir eine beliebige Farbe aus. Nun kannst du die von Hand abgewickelte Textur des Roboters oder den Roboter direkt bemalen. | 120501  Aufgabe wird gesprochen (über das i kann sich der Nutzer die Aufgabe anzeigen lassen)  Klickt der Nutzer auf die Textur, vergrößert sich diese  Klickt der Nutzer auf den Roboter, vergrößert sich dieser  Roboter kann vom Nutzer beliebig über das Steuerkreuz gedreht und von allen Seiten betrachtet werden |

**12.6 (A) Mip-Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 120601  Das Mip-Mapping ist eine Anti-Aliasing-Methode. In der Computergrafik tritt der Alias-Effekt beim Betrachten von einer Textur mit hoher Auflösung aus weiter Entfernung auf, wodurch Muster und Verzerrungen entstehen, die im Originalbild nicht enthalten sind.  120602  Beim Mip-Mapping werden von einer Textur mehrere vorberechnete skalierte Texturen mit sinkender Auflösung berechnet. Dabei wird in jedem Schritt die Kantenlänge des Originals halbiert.  120603  Ist das texturierte Polygon nahe beim Betrachter oder groß skaliert, kommt eine große Textur mit hoher Auflösung zum Einsatz. Ist es jedoch weiter entfernt oder klein skaliert, wird eine kleinere Textur mit geringerer Auflösung verwendet.  120604  Der große Vorteil des Mip-Mappings besteht darin, das die verschiedenen Maps zum Zeitpunkt des Renderns bereits vorberechnet sind. | * Anti-Aliasing-Methode * Entstehung von Verzerrungen und ungewollten Mustern * Kantenlänge wird bei jedem Schritt halbiert * Objekt nahe beim Betrachter: große Textur mit hoher Auflösung * Objekt weiter entfernt vom Betrachter: kleine Textur mit niedriger Auflösung | 120601  Einblenden eines Gitters mit Alias Effekt  120602  Einblenden einer großen Map mit hoher Auflösung, nahe beim Betrachter  Einblenden einer zweiten Map mit halbierten Kantenlängen  120603  Einblenden einer dritten Map mit nochmals halbierten Kantenlängen  Einblenden des Fluchtpunktes als gestrichelte Linien  120604  Einblenden des Gitters mit Mip-Mapping und ohne Alias-Effekt |

Grafikablauf:

**   ** 

**12.7 (A) Bump-Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 120701  Eine normale 2D-Textur verleiht einem Objekt keine Oberflächeneigenschaften wie Höhe und Tiefe. Das Bump-Mapping dient dazu, einem 3D-Objekt diese Eigenschaften zu verleihen, ohne dabei die Geometrie des 3D-Objektes zu beeinflussen.  120702  Beim Bump-Mapping wird auf Pixelebene lediglich die Normale der Oberfläche verändert. Es wird mit Hilfe von Schattierung und Reflexion eine Illusion von Tiefe auf dem 3D-Objekt erzeugt. Das Bump-Mapping verwendet in der Regel verschiedene Graustufen und beschreibt damit Höhenunterschiede auf dem Objekt. Somit ist das Bump-Mapping ein reiner Beleuchtungseffekt.  120703  Der Vorteil des Bump-Mapping besteht darin, dass bei diesem Mapping-Verfahren der Speicherplatz und die Rendering-Zeit ziemlich gering bleiben. | * Erzeugung von Tiefe ohne Beeinflussung der Geometrie * Veränderung der Normalen der Oberfläche * Illusion durch Graustufen * Beleuchtungseffekt * Vorteil: geringe Erhöhung des Speicherplatzes und der Rendering-Zeit | 120701  Einblenden einer Sphäre mit Bump Mapping  Sphäre dreht sich um sich selbst |

**12.8 (A) Normal-Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 120801  Das Normal-Mapping ist eine Abwandlung des Bump-Mapping-Verfahrens. Es zielt ebenfalls darauf ab, einen größeren Detailreichtum eines Objektes in Form von Höhe und Tiefe zu erzeugen.  120802  Beim Normal-Mapping wird die Anzahl der Polygone nicht erhöht und die Oberfläche nur optisch verformt. Alle Informationen über die Ausrichtung der Normalen, die für die Beleuchtung wichtig sind, werden in Form von RGB-Werten von einem hoch aufgelösten auf ein niedrig aufgelöstes Objekt übertragen. Die optischen Details gehen dabei nicht verloren.  120803  Der Nachteil dieses Mapping-Verfahrens ist, dass es stark vom Betrachtungswinkel abhängig ist. | * Abwandlung des Bump-Mapping * Ziel: größeren Detailreichtum * Illusion durch Übertragung von RGB-Werten * optische Details bleiben erhalten * Nachteil: winkelabhängig | 120801  Einblenden einer Sphäre mit Normal-Mapping  Sphäre dreht sich um sich selbst |

**12.9 (A) Displacement-Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 120901  Das Displacement-Mapping löst dasselbe Problem wie das Bump- und das Normal-Mapping, es verleiht dem 3D-Objekt ebenfalls Vertiefungen und Erhöhungen.  120902  Der Unterschied zwischen den Mapping-Verfahren ist, dass beim Displacement Mapping die Geometrie des 3D-Objektes verändert wird. Das Displacement-Mapping findet also auf polygonaler Ebene statt.  120903  Durch “Verschieben” beziehungsweise “Verdrängen” des Materials werden dem 3D-Objekt die gewünschten Oberflächeneigenschaften verliehen. | * Vertiefungen und Erhöhungen der Oberfläche des Objektes * Veränderung der Form durch Veränderung der Geometrie des 3D-Objektes * displacement = “verschieben” bzw. “verdrängen” des Materials | 120901  Einblenden einer Sphäre mit Displacement-Mapping  Sphäre dreht sich um sich selbst |

**12.10 (I) Vergleich**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 121001  Hier wird der Unterschied zwischen dem Normal-Mapping und dem Displacement-Mapping dargestellt. Betrachte die texturierten 3D-Objekte von allen Seiten. | Hier wird der Unterschied zwischen dem Normal-Mapping und dem Displacement-Mapping dargestellt. Betrachte die texturierten 3D-Objekte von allen Seiten. | 121001  Aufgabe wird gesprochen (über das i kann sich der Nutzer die Aufgabe anzeigen lassen)  Auswahlmöglichkeiten  “Normal-Mapping”  “Displacement-Mapping”  Der Nutzer kann sich das Objekt über das Steuerkreuz von allen Seiten ansehen  (Eventuell Intensitätsregler, über den der Nutzer die Intensität des jeweiligen Mapping-Verfahrens beliebig anpassen kann) |

Grafikablauf:

** **

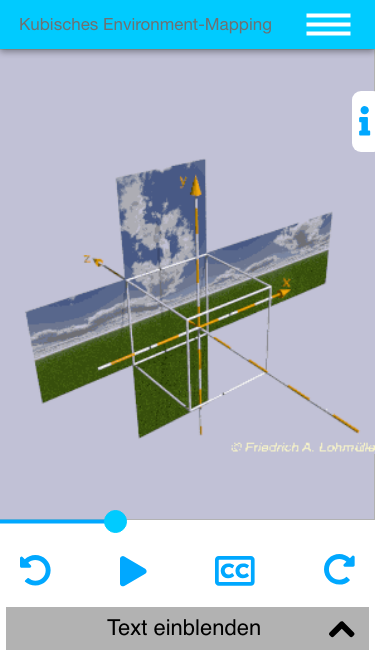
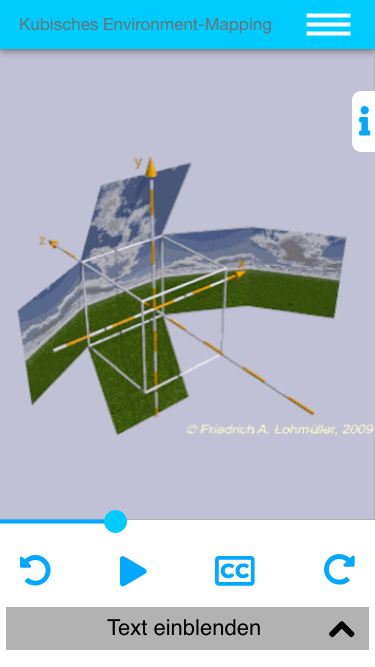
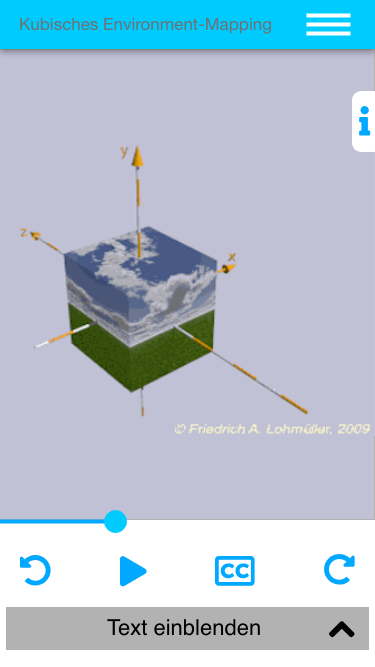
**12.11 (A) Environment-Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 121101  Mit dem Environment-Mapping werden in der Computergrafik Spiegelungen, durchlässige Objekte und Beleuchtungs-Simulationen dargestellt.  121102  Dabei wird die Umgebung, bestehend aus Objekten und Lichtquellen, in der sich das Objekt befinden soll, als eine Umgebungs-Textur gespeichert und auf das 3D-Objekt projiziert. Wie dieser Vorgang funktioniert erfährst du im nächsten Kapitel. Das 3D-Objekt muss im Verhältnis zu seiner Umgebung eher klein sein, damit die Spiegelung realistisch wirkt.  121103  Beim Environment-Mapping wird in zwei Arten unterschieden: das sphärische und das kubische Environment-Mapping. In Folge wird nur Letzteres beschrieben. | * Simulation von spiegelnden Objekten * Speichern einer Umgebungstextur (umliegende Objekte und Lichtquellen) * Sphärisches Environment-Mapping * Kubisches Environment- Mapping | 121101  Einblenden eines spiegelnden 3D-Objektes (hier: Seifenblase, später: spiegelnder Roboter) |

**12.12 (A) Kubisches Environment-Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| **Finaler Screen** | **Sprechertext** | **Screentext (i)** | **Regieanweisung** |
|  | 121201  Bei dem kubischen Environment-Mapping wird die Umgebung auf einem Kubus abgebildet.  121202  Es werden sechs 2D-Umgebungs-Texturen verwendet, die zusammen die Fläche eines Kubus bilden.  121203  Die sechs einzelnen Umgebungs-Texturen entstehen durch das Fotografieren oder Rendern aus Position des Objektmittelpunkts in einem Winkel von 90 Grad. Dabei müssen die sechs Würfelflächen randlos abgedeckt werden. | * Kubisches Environment- Mapping: Abbildung der Umgebung auf einen Kubus * Vorteil: keine Verzerrung * Verwendung von sechs 2D-Umgebungs-Texturen | 121201  Einblenden von sechs 2D-Umgebungs-Texturen  Die sechs 2D-Umgebungs-Texturen legen sich um den Kubus und decken alle Flächen randlos ab  2D-Textur klappt wieder auf  Vorgang wiederholt sich, solanger der Sprecher spricht  Quelle und Code-Beispiel: Friedrich A. Lohmüller, http://www.f-lohmueller.de/pov\_tut/backgrnd/p\_sky9d.htm |

Grafikablauf:

**   **