財団法人 デジタルコンテンツ協会
Digital Content Association of Japan

デジタルコンテンツ制作の先端技術応用に関する調査研究

「ゲーム開発技術の歴史と展望」

三宅　陽一郎
（株式会社フロム・ソフトウェア）

y.m.4160@gmail.com

2010.3.19

Contact Information

Youichiro Miyake

- Mail: y.m.4160@gmail.com
- Twitter: @miyayou
- Blog: http://blogai.igda.jp
- LinkedIn: http://www.linkedin.com/in/miyayou
- Facebook: http://www.facebook.com/youichiro.miyake

目次

第一部
なぜゲーム開発技術の歴史を編纂するのか？

ゲーム開発技術の歴史

（2007年度の時点）

ゲーム開発技術の拡大と専門化が加速する
殆ど文献がなく、整備されていない。
70年代後半から活躍された先人が引退を始める。

過去を持たないものは未来を持つことはできない

（2008年度、2009年度）

全技術分野のエキスパートに執筆依頼。

①過去30年分の歴史的経緯　②年表　③これからの展望

せめて10年置きにこういった調査をしておいてくれていたら…
この報告書はこれから10年単位で役に立つ仕事。

ゲーム開発技術ロードマップ

方針：　ゲーム開発技術の歴史を編纂する

これまで殆ど編纂されることがなかった
ゲーム開発技術の歴史を編纂し、
現在までの日本のゲーム開発技術の系譜を明らかにし、
これからの展望へ繋げて行く

期間：　2008年4月～2010年3月

実施方法：

① 各分野のエキスパートに歴史的見地から執筆依頼
② インタビュー

目標：　ゲーム産業の技術をゲーム産業自身が明らかにする

① 技術戦略の足場を固める。
② ゲーム産業の技術の歴史を明らかにする。
③ ゲーム産業の技術者の技術的位置と立場を明らかにする。
④ ゲーム産業の新人にこれまでの歴史を解説する。
⑤ 複数の視点から歴史を検証する

執筆者の皆さん（21分野21人）

ゲームデザイン　井上明人(国際大学GLOCOM研究員)

3D CG製作　川島 基展(東京工科大学 片柳研究所クリエイティブ・ラボ)

サウンド製作　細江　慎治（株式会社スーパースィープ）

アニメーション製作　金久保　哲也(株式会社バンダイナムコゲームス)

プログラミングAI　三宅　陽一郎（株式会社フロム・ソフトウェア）

プログラミング　グラフィックス描画　宮澤 篤　大久保 明（株式会社バンダイナムコゲームス）

プログラミング　物理・衝突判定　長谷川　晶一（電気通信大学）

タスクシステム技術　田村　祐樹　（株式会社ネバーランドカンパニー）　大野　功二（オープランニング）
黒須一雄（株式会社モバイル＆ゲームスタジオ）

プログラミング　スクリプト　小久保　啓三（専門学校HAL東京）

ネットワーク通信　佐藤カフジ（フリーライター）

開発方法論　長久　勝　（ハイパーコンテンツ(株)/早稲田大学メディアネットワークセンター）

2Dグラフィックス　近藤敏信氏（有限会社スタジオ）

3Dグラフィックス　シェーダー　今給黎隆　(株式会社バンダイナムコゲームス)

ゲーム物理シミュレーション　今給黎隆　(株式会社バンダイナムコゲームス)

ミドルウェア　佐藤カフジ　（フリーライター）

シナリオ　すずきこういち　（シナリオライター）

ユーザーインターフェイス（ハードウェア）　大神佳人氏（株式会社HORI）

ユーザーインターフェイス（ソフトウェア）　サイトウアキヒロ氏（立命館大学）

開発工程管理　田村　祐樹　（株式会社ネバーランドカンパニー）

ローカライズ　長谷川亮一　（セガ）

アセンブラ　山口　誠　（株式会社グッド・フィール）

デジタルコンテンツ制作の
先端技術応用に関する調査研究報告書

2007年度 （300ページ）
http://www.dcaj.org/report/2007/data/dc08_07.pdf
2008年度 （400ページ）
http://www.dcaj.org/report/2008/data/dc_08_03.pdf
2009年度 （400ページ）
http://www.dcaj.org/report/2009/data/dc_09_03.pdf

全技術分野のエキスパートの原稿
①過去30年分の歴史的経緯（～15ページ程度）
②年表
③これからの展望

原稿例

する手法である。2 体問題に帰着できるので計算が簡単になるが、短時間のシミュレーションに非常に多くの衝突が発生する状況が起こりえるため、リアルタイムシミュレーションには向かない。また関節や床の上に積み重なった物体のような継続した拘束を扱うことも出来ない。

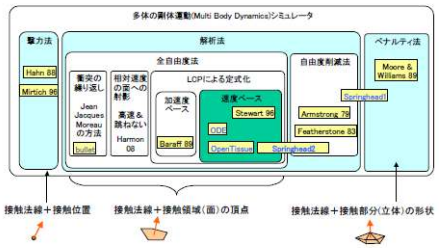


図 3.2-90　接触力計算法によるシミュレータの分類と必要な接触情報

(b)　接触判定

接触力の計算をするためには、まずどの物体がどの物体とどこでどう接触しているのかを調べる必要がある。接触判定はまず大まかに大域的な判定をおこない、接触している可能性がある物体のペアを求め(大域判定)、次に各ペアについて本当に接触しているのか、どのように接触しているのかを調べる(詳細判定)。

① 大域判定の手法

大域判定では、Bounding Volume とよばれる球や直方体のような簡単な形状で物体を囲み、Bounding Volume 同士が接触しているかどうか判定する。

また、時間がかかる総当り判定を避けることによる高速化も行われる。ゲーム世界に n 個の物体があるとき、総当りで判定を行うと n(n-1)/2 のペアについて判定を行うことになる。このため計算量は O(n²)になる。空間を 2 つに分割し、どちらの空間に含まれるかにより n 個の物体を 2 つのグループに分割すれば（両方に含まれる物体は両方のグルー

コンテンツデータを埋め込むことであった。ここで埋め込まれたデータは、固定のフォーマットのものから、徐々に、より自由度の高い、一定の取り決めのもとに実行される中間コードのような形態に進化していく。ここにおいて、その中間コードを解釈、実行する処理系(VM=バーチャルマシン)が、存在し始めたことになる。既存のマクロアセンブラを利用して、中間コードを生成し、実行時にそれを解釈するという、ごくごく原始的なスクリプトシステムが、生まれたと言えよう。

実際、ファミコンやスーパーファミコンにおいて、長いストーリー展開を持つロールプレイングゲーム(ドラゴンクエストやファイナルファンタジー)は、当初、このような形態でコンテンツが記述されていた。

その後、中間コードを生成するにあたって、既存のマクロアセンブラを利用するのではなく、独自の変換プログラム(コンパイラ)を用いるように変わってきた。コンテンツ記述の柔軟性と生産性を、より上げるためである。

スクリプトシステムが誕生とともに、成長していった時代である。

[当時のスクリプト言語(というよりデータ記述)の一例]

村人移動データ:

```
座標 X1,座標 Y1,10      ；フレーム
座標 X2,座標 Y2,10      ；フレーム
座標 X3,座標 Y3,10      ；フレーム
0                                    :END
```

村人メッセージデータ:

```
DATA　ひがしのやまには、まものがでる
0       : END
```

(5)　大容量ゲーム機(プレイステーション)の時代

ゲーム機に大容量の CD-ROM メディアが使われるようになり、また、メモリや実行スピードにも、ある程度の余裕が生まれるようになり、従来のマクロアセンブラを利用するようなコンテンツ記述はなくなった。RISC チップの構成により、システムのプログラミングそのものも、記述性の高い C 言語に置き換わることになる。もちろん、C のプリプロセッサを駆使して、コンテンツ記述を、C 言語に展開することも可能で、比較的小規模のゲーム・コンテンツでは、そうした手法がとられることもある。大勢においては、各ゲームメーカーが、それぞれのジャンルのゲームの特性に即した独自の言語を開発し、それを中間コードにコンパイルし、実行時にバーチャルマシンでコードを解釈しながら、ゲームを制御するような手法をとっていた。この時代、メーカー内でのスクリプトシステムが、確固とした基盤を築いていく。

仕様を満たしているか、要求を満たしているか、を順に試験する。実装上の不具合や設計を充足していない場合は実装工程に、そこで不具合が見つからず、仕様を満たしていないことが分かった場合は設計工程に、更にそこで不具合が見つからず、要求を満たしていない場合は分析・仕様化工程に、そこで不具合がなければ要求獲得工程に、それぞれさかのぼって変更・修正作業を行う。

さかのぼって変更・修正した場合、当然、それ以降の工程も影響を受けるため、順次作業が必要となる。これは、要求獲得工程の変更・修正を行った場合、それ以降の仕様策定、設計、実装、全ての工程で作業が発生することを意味する。ここで大きな問題となるのが、既に試験をパスした工程で変更・修正が生じるため、試験も全てやり直しとなることである。要求の不具合が見つかるのは、実装、設計、仕様の試験をパスした後なので、要求の変更・修正に紐付いて、仕様、設計、実装の変更・修正が生じ、パスした試験に再度挑むこととなる。

この問題に対しては、事前の試験設計と試験の自動化による回帰試験の効率化や、疎結合なモジュール設計による変更範囲の局地化などの対策が取れる。

また、変更管理が機能すれば、変更・修正の影響が追跡可能となり、影響範囲を局地的に見られるようになるため、各工程の成果物のレビュー精度が上がる。レビューのコストが下がれば、試験ではなくレビューで不具合を検出することも対策となり得る。こうした観点から、試験では実装のバグのみを対象とし、設計以前の工程の不備は、それぞれの工程、もしくは直後の工程でレビューによって発見し、即座に変更・修正を行う開発プロセス「動的 V 字モデル」が提唱されている。動的 V 字モデルは、工程間を密につなぎ、定常的なフィードバックを行うため、ウォーターフォール型ではない。

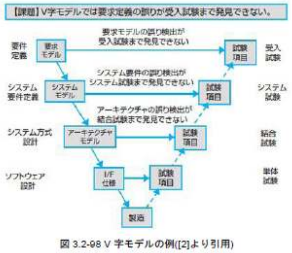


図 3.2-98 V 字モデルの例([2]より引用)

クリメンタルな反復型と、初期の繰り返しをプロトタイピングと位置付け、文書化などの作業を軽減した Boehm のスパイラル型が現れた。

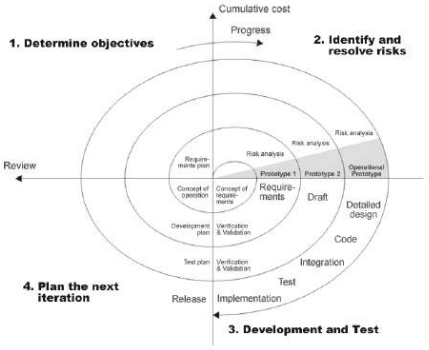


図 3.2-100　Boehm のスパイラル型の模式図([5]より引用)

(3)　PDC(S)A

反復型開発において、何を行うか計画し(Plan)、その計画に基づいて行動し(Do)、その結果を吟味し(Check)、改善を行う(Action)、4 つの行動を繰り返すサイクルを、PDCA サイクルと呼ぶ。PDCA サイクルは、モノ作りの現場への洞察から生まれたが、今日では、サービス業など、モノ作り以外でも広く用いられている、改善活動の枠組みである。Walter A. Shewhart の示した概念を W. Edwards Deming が整理し、1939 年に出版した「Statistical method from the viewpoint of quality control」が、その原点と言われている。

Shewhart は、AT&T ベル研究所に設立当初から所属していた、統計的品質管理の第一人者である。Shewhart は、ベル研究所所属以前に、機器製造の検査技術部門で、管理図

原稿例

する手法である。2 体問題に帰着できるので計算が簡単になるが、短時間のシミュレーションに非常に多くの衝突が発生する状況が起こりえるため、リアルタイムシミュレーションには向かない。また関節や床の上に積み重なった物体のような継続した拘束を扱うことも出来ない。

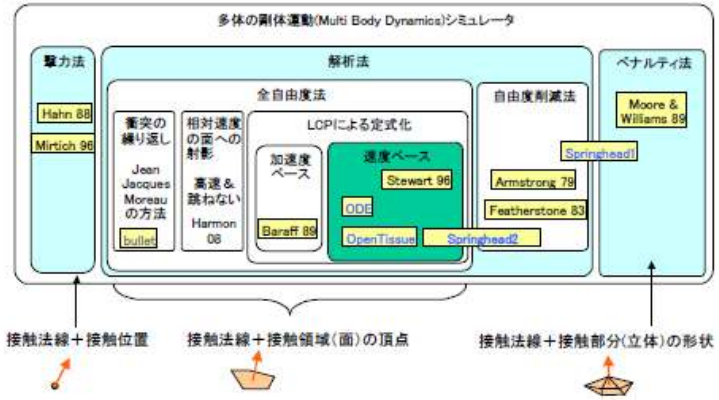


図 3.2-90　接触力計算法によるシミュレータの分類と必要な接触情報

(b)　接触判定

接触力の計算をするためには、まずどの物体がどの物体とどこでどう接触しているのかを調べる必要がある。接触判定はまず大まかに大域的な判定をおこない、接触している可能性がある物体のペアを求め(大域判定)、次に各ペアについて本当に接触しているのか、どのように接触しているのかを調べる(詳細判定)。

① 大域判定の手法

大域判定では、Bounding Volume とよばれる球や直方体のような簡単な形状で物体を囲み、Bounding Volume 同士が接触しているかどうか判定する。

また、時間がかかる総当り判定を避けることによる高速化も行われる。ゲーム世界に n 個の物体があるとき、総当りで判定を行うと n(n-1)/2 のペアについて判定を行うことになる。このため計算量は $O(n^2)$になる。空間を 2 つに分割し、どちらの空間に含まれるかにより n 個の物体を 2 つのグループに分割すれば（両方に含まれる物体は両方のグルー

コンテンツデータを埋め込むことであった。ここで埋め込まれたデータは、固定のフォーマットのものから、徐々に、より自由度の高い、一定の取り決めのもとに実行される中間コードのような形態に進化していく。ここにおいて、その中間コードを解釈、実行する処理系(VM=バーチャルマシン)が、存在し始めたことになる。既存のマクロアセンブラを利用して、中間コードを生成し、実行時にそれを解釈するという、ごくごく原始的なスクリプトシステムが、生まれたと言えよう。

実際、ファミコンやスーパーファミコンにおいて、長いストーリー展開を持つロールプレイングゲーム(ドラゴンクエストやファイナルファンタジー)は、当初、このような形態でコンテンツが記述されていた。

その後、中間コードを生成するにあたって、既存のマクロアセンブラを利用するのではなく、独自の変換プログラム(コンパイラ)を用いるように変わってきた。コンテンツ記述の柔軟性と生産性を、より上げるためである。

スクリプトシステムが誕生とともに、成長していった時代である。

[当時のスクリプト言語(というよりデータ記述)の一例]

村人移動データ:

```
座標 X1,座標 Y1,10      ; フレーム
座標 X2,座標 Y2,10      ; フレーム
座標 X3,座標 Y3,10      ; フレーム
0                                    :END
```

村人メッセージデータ:

```
DATA  ひがしのやまには、まものがでる
0     ; END
```

(5)　大容量ゲーム機(プレイステーション)の時代

ゲーム機に大容量の CD-ROM メディアが使われるようになり、また、メモリや実行スピードにも、ある程度の余裕が生まれるようになり、従来のマクロアセンブラを利用するようなコンテンツ記述はなくなった。RISC チップの構成により、システムのプログラミングそのものも、記述性の高い C 言語に置き換わることになる。もちろん、C のプリプロセッサを駆使して、コンテンツ記述を、C 言語に展開することも可能で、比較的小規模のゲーム・コンテンツでは、そうした手法がとられることもある。大勢においては、各ゲームメーカーが、それぞれのジャンルのゲームの特性に即した独自の言語を開発し、それを中間コードにコンパイルし、実行時にバーチャルマシンでコードを解釈しながら、ゲームを制御するような手法をとっていた。この時代、メーカー内でのスクリプトシステムが、確固とした基盤を築いていく。

原稿例

仕様を満たしているか、要求を満たしているか、を順に試験する。実装上の不具合や設計を充足していない場合は実装工程に、そこで不具合が見つからず、仕様を満たしていないことが分かった場合は設計工程に、更にそこで不具合が見つからず、要求を満たしていない場合は分析・仕様化工程に、そこで不具合がなければ要求獲得工程に、それぞれさかのぼって変更・修正作業を行う。

さかのぼって変更・修正した場合、当然、それ以降の工程も影響を受けるため、順次作業が必要となる。これは、要求獲得工程の変更・修正を行った場合、それ以降の仕様策定、設計、実装、全ての工程で作業が発生することを意味する。ここで大きな問題となるのが、既に試験をパスした工程で変更・修正が生じるため、試験も全てやり直しとなることである。要求の不具合が見つかるのは、実装、設計、仕様の試験をパスした後なので、要求の変更・修正に紐付いて、仕様、設計、実装の変更・修正が生じ、パスした試験に再度挑むこととなる。

この問題に対しては、事前の試験設計と試験の自動化による回帰試験の効率化や、疎結合なモジュール設計による変更範囲の局地化などの対策が取れる。

また、変更管理が機能すれば、変更・修正の影響が追跡可能となり、影響範囲を局地的に見られるようになるため、各工程の成果物のレビュー精度が上がる。レビューのコストが下がれば、試験ではなくレビューで不具合を検出することも対策となり得る。こうした観点から、試験では実装のバグのみを対象とし、設計以前の工程の不備は、それぞれの工程、もしくは直後の工程でレビューによって発見し、即座に変更・修正を行う開発プロセス「動的 V 字モデル」が提唱されている。動的 V 字モデルは、工程間を密につなぎ、定常的なフィードバックを行うため、ウォーターフォール型ではない。

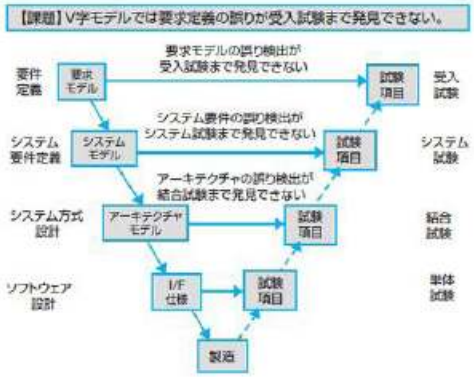


図 3.2-98 V 字モデルの例([2]より引用)

クリメンタルな反復型と、初期の繰り返しをプロトタイピングと位置付け、文書化などの作業を軽減した Boehm のスパイラル型が現れた。

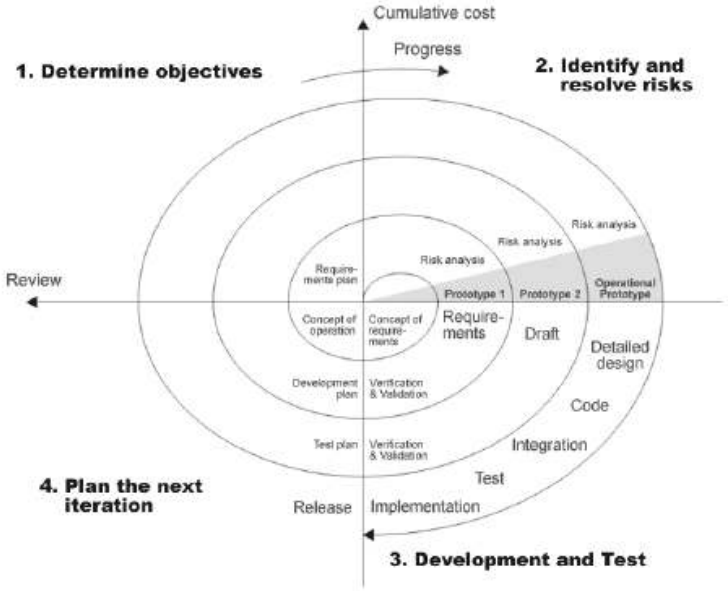


図 3.2-100　Boehm のスパイラル型の模式図([5]より引用)

(3)　PDC(S)A

反復型開発において、何を行うか計画し(Plan)、その計画に基づいて行動し(Do)、その結果を吟味し(Check)、改善を行う(Action)、4 つの行動を繰り返すサイクルを、PDCA サイクルと呼ぶ。PDCA サイクルは、モノ作りの現場への洞察から生まれたが、今日では、サービス業など、モノ作り以外でも広く用いられている、改善活動の枠組みである。Walter A. Shewhart の示した概念を W. Edwards Deming が整理し、1939 年に出版した「Statistical method from the viewpoint of quality control」が、その原点と言われている。

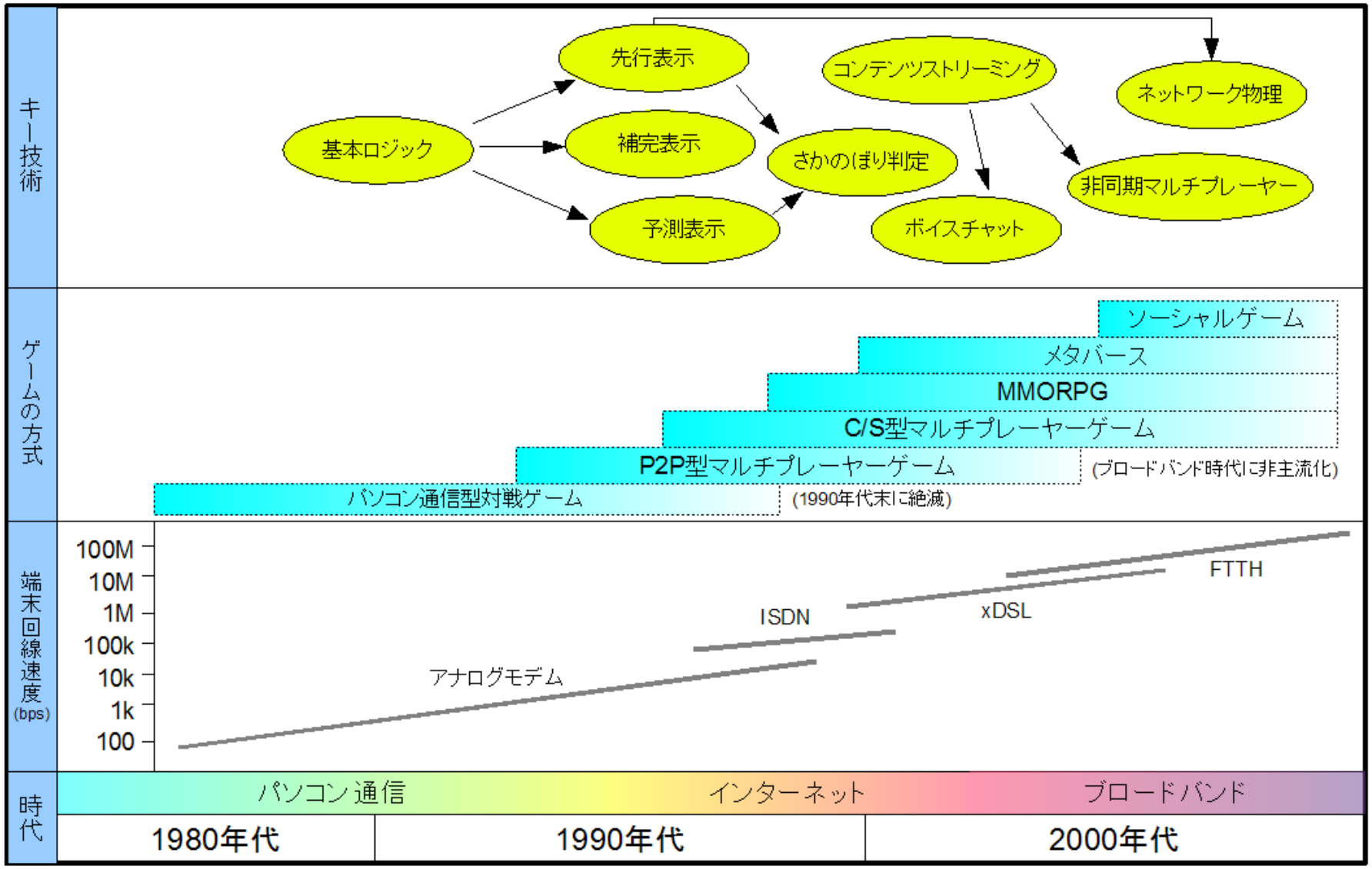
Shewhart は、AT&T ベル研究所に設立当初から所属していた、統計的品質管理の第一人者である。Shewhart は、ベル研究所所属以前に、機器製造の検査技術部門で、管理図

年表例①

表 3.2-06　スクリプティングの歴史

（2008年度報告書 「スクリプティングの歴史」 大久保）

年表例②



（2008年度報告書「ネットワーク技術」 佐藤）

年表例③

表 3.2-07　関連年表

年	出来事、論文、刊行物など
1939	Walter A. Shewhart, W. Edwards Deming "Statistical method from the viewpoint of quality control"
1950	Deming が日本に PDC(S)A を紹介
1970	Winston W. Royce "Managing the development of large software systems"
1985	DOD-STD-2167
1986	野中 郁次郎, 竹内 弘高 "The New New Product Development Game"
1988	Barry Boehm "A Spiral Model of Software Development and Enhancement"
1990	James Womack, Daniel Jones, Daniel Roos "The Machine That Changed the World"
1993	Jeff Sutherland、John Scumniotales、Jeff McKenna、Ken Schwaber らが Scrum を始める
1996	Kent Beck が XP(Extreme Programming)を考案
1997	Valve 社が Half-Life の開発に Cabal を適用
2001	アジャイル宣言

（2008年度報告書「開発方法論」長久）

年表例④

(6)　タスクシステムの歴史年表

年	出来事
1979	ギャラクシアン(ナムコ)発売。後にタスクシステムを最初に使ったゲームとの推測がインターネット上で広がる。 この時期にタスクシステムの原型となるシステムがゲーム業界において開発されたと推測される。
1983	ゼビウス(ナムコ)発売。ファミリーコンピューター(任天堂)発売。
1990	スーパーファミコン(任天堂)発売
1991	ストリートファイターⅡ(カプコン)発売。格闘ゲームがブームになる。
1994	SEGA Saturn (セガ・エンタープライズ)発売。 Play Station(ソニー・コンピュータエンタテイメント)発売。
1996	セガサターン版「NOON」(大野 功二/マイクロキャビン)発売。
1999	Dreamcast(セガ・エンタープライズ)発売。 「簡単ゲーム工房」(大野 功二/Country Fox:工学社)発売
2000	Play Station2(ソニー・コンピュータエンタテイメント)発売。 「Logician Lord」(Peto 氏)の WEB サイトで、タスクシステムの解説を更新。
2001	「White Paper(ほわいと・ぺーぱー)」(Moha 氏)の WEB サイトで、タスクシステムの解説を更新。
2003	「Windows プロフェッショナルゲームプログラミング 2」(やねうらお氏: 秀和システム)でタスクシステムの解説を発表。 「最新 DirectX 実践テクニック　Part3 Managed DirectX とタスクシステムを使ったゲーム・プログラミング」(C MAGAZINE 2003 年 12 月号掲載　本多直人氏: ソフトバンク クリエイティブ) でタスクシステムの解説を発表。
2004	「ゲームのためのタスクシステム」(C MAGAZINE 2004 年 12 月号掲載　松浦健一郎氏: ソフトバンク クリエイティブ) でタスクシステムの解説を発表。
2005	Xbox360(マイクロソフト)発売。
2006	PLAYSTATION3(ソニー・コンピュータエンタテイメント)発売。 CEDEC 2006「次世代機に向けたゲームエンジンの設計」(石田 智史氏:株式会社カプコン)にて、MT Framework を発表。 「ゲーム・ノ・シクミ シューティングゲーム編 第 11 回 C++によるタスクシステムの実現」(C MAGAZINE 2006 年 1 月号掲載　松浦 健一郎氏/司 ゆき氏: ソフトバンク クリエイティブ)、「シューティングゲーム プログラミング 」(2006 年 9 月発売　松浦 健一郎氏 / 司ゆき氏: ソフトバンククリエイティブ) でタスクシステムの解説を発表。

（2008年度報告書 「タスクシステム」 大野）

参考文献

日本デジタルゲーム学会 公開講座

武田寧「2000年代・現在の環境とHSP、プログラミング教育について」
http://www.digrajapan.org/modules/mydownloads/images/study/20091127_90_00.pdf

水上恵太「80年代・90年代のゲーム開発環境について」
http://www.digrajapan.org/modules/mydownloads/images/study/20091127_80_90.pdf

武田寧「70年代・80年代のゲーム開発環境について」
http://www.digrajapan.org/modules/mydownloads/images/study/20091127_70_80.pdf

日本デジタルゲーム学会 学会誌　デジタルゲーム学研究 第3巻1号

コンピュータゲームのテクノロジー、再論 （宮澤 篤・大久保 明）
ゲーム映像の変遷と未来―レースゲームに見るゲームデザイン序論―（中村 勲）
デジタルゲームにおけるアニメーション制作事例
―モーションキャプチャーからフェイシャルアニメーションまで―
（柳原孝安）

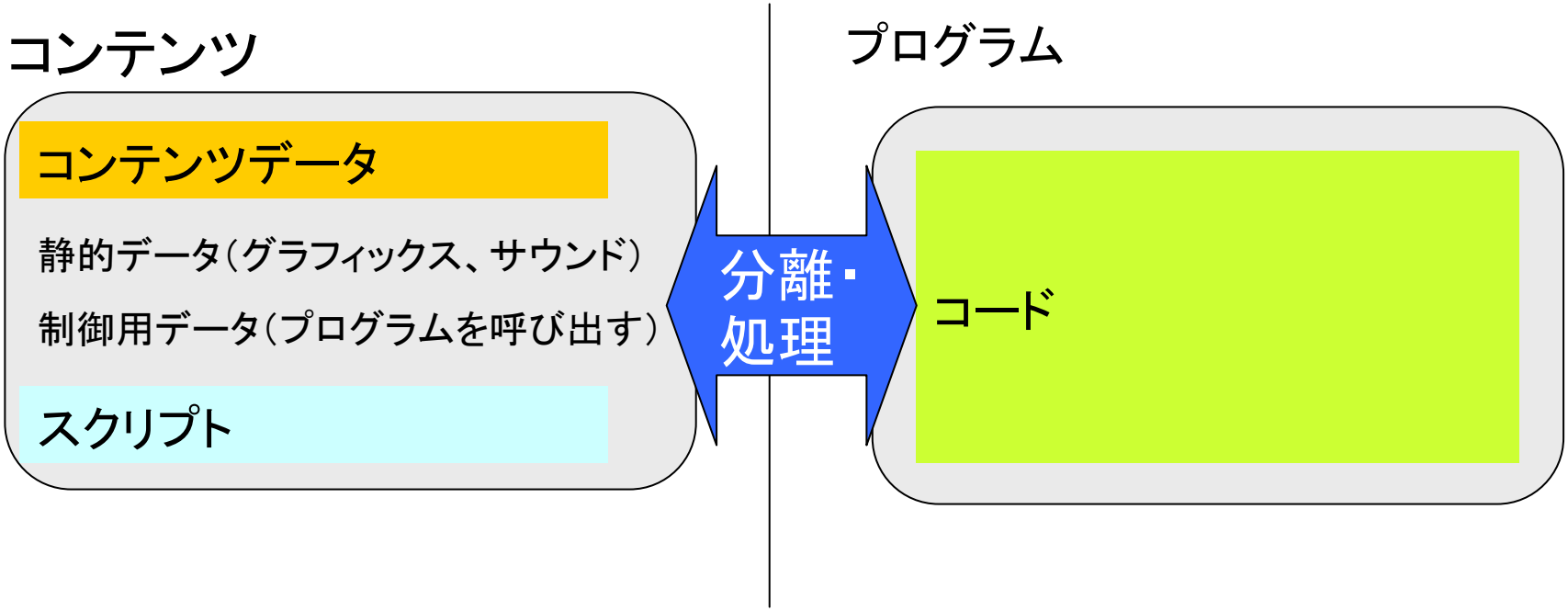
参考文献

IGDA日本 インディーズゲーム専門部会　第6回研究会

「同人シューティングゲームの潮流
- 80年代から現代まで、変化から学ぶ実践的な普遍性 - 」
http://www.slideshare.net/sakugetu

第二部　ゲーム開発技術の歴史概要

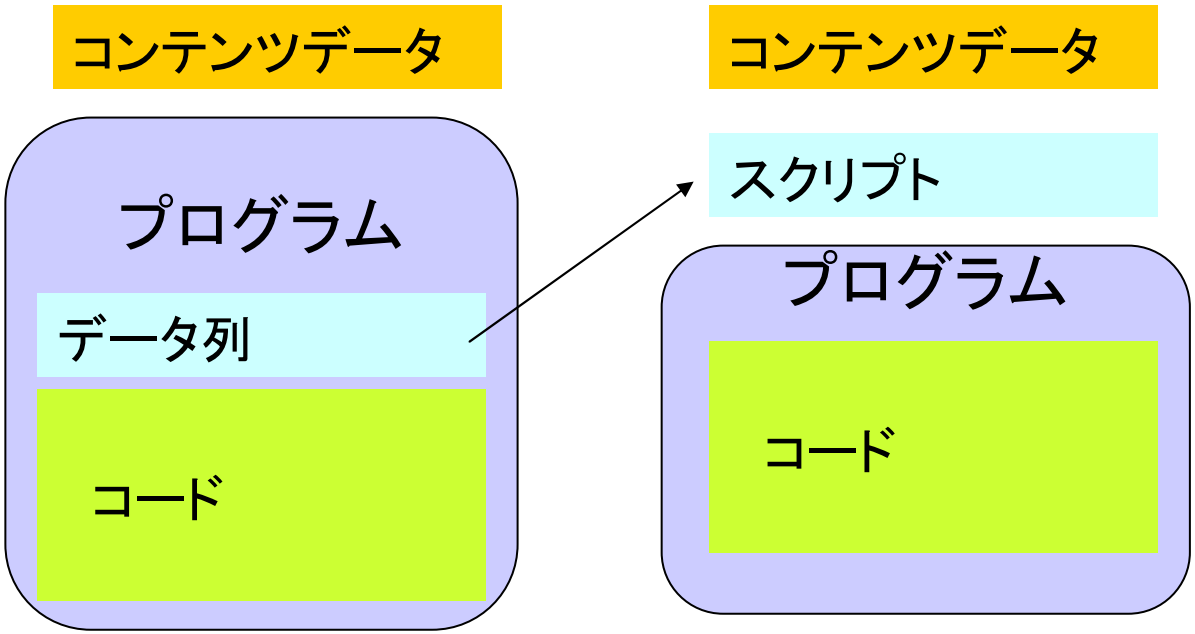
ゲームデータの3つの要素



この初期の段階で、なんらか、制御プログラムと表示データの分離、
物語の制御コードもデータとして分離するようなプログラミングが行われていた。

スクリプトのもととなるものは、「データ列」として、プログラム中に混在していた時代が、長く続く。
比較的小規模なゲームプログラムであれば、これは今も同じだろう。（小久保）

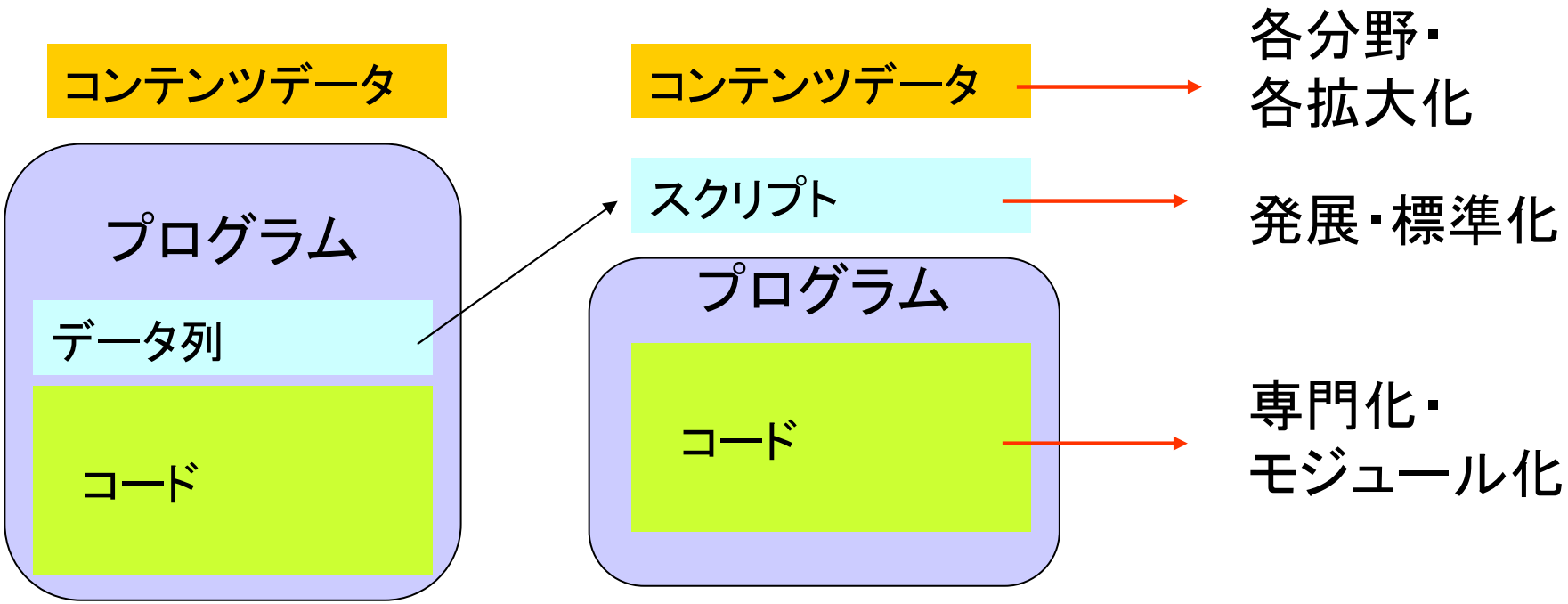
1970 年代	1980 年代	1990 年代前半	1990 年代後半	2000 年代前半	2000 年代後半

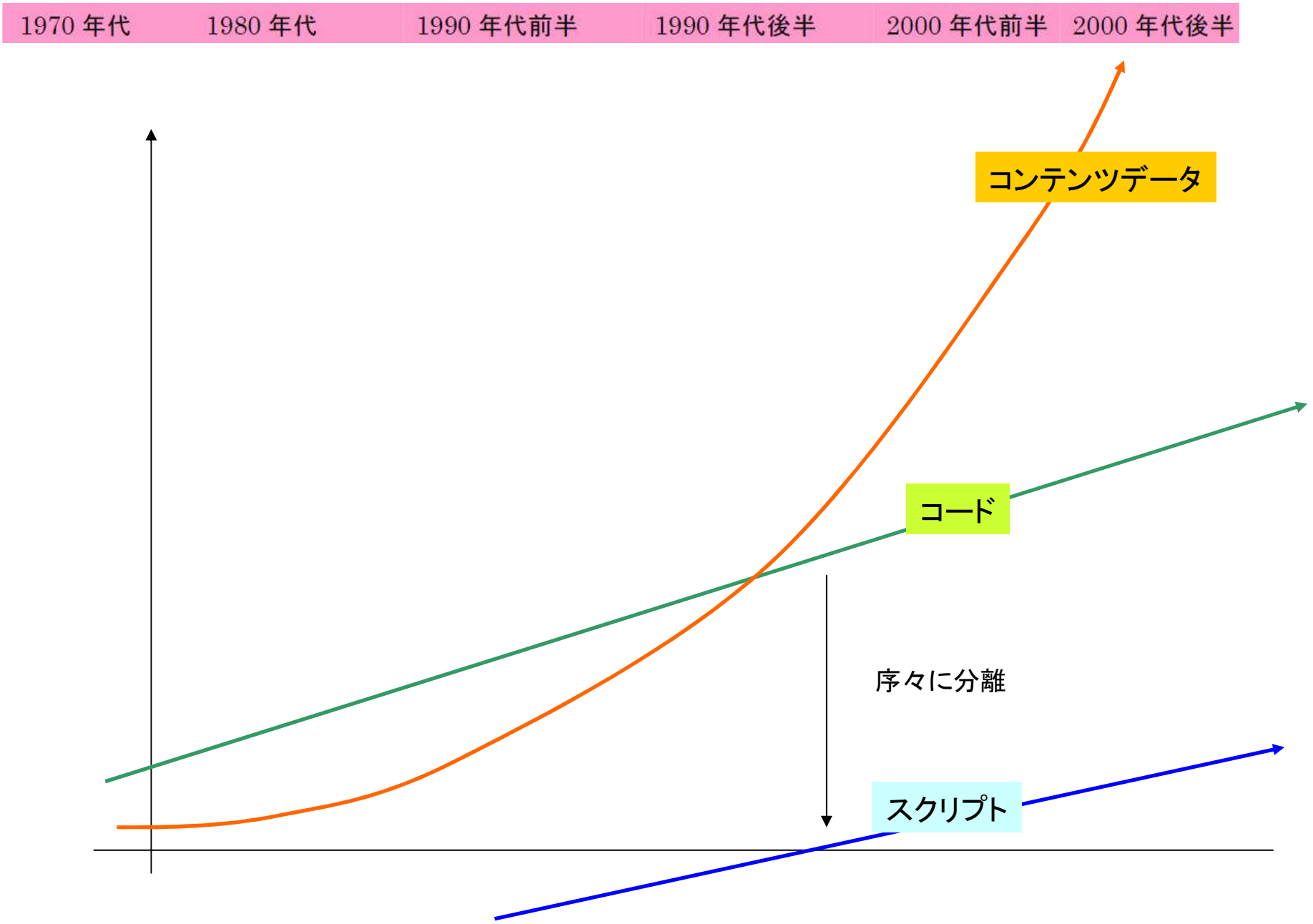


この初期の段階で、なんらか、制御プログラムと表示データの分離、
物語の制御コードもデータとして分離するようなプログラミングが行われていた。

スクリプトのもととなるものは、「データ列」として、プログラム中に混在していた時代が、長く続く。
比較的小規模なゲームプログラムであれば、これは今も同じだろう。（小久保）

1970 年代	1980 年代	1990 年代前半	1990 年代後半	2000 年代前半	2000 年代後半




1970 年代
1980 年代
1990 年代前半
1990 年代後半
2000 年代前半
2000 年代後半
コンテンツデータ
コード
序々に分離
スクリプト

ゲームデータの3つの要素による執筆3分野

3D CG製作　川島 基展(東京工科大学 片柳研究所クリエイティブ・ラボ)
サウンド製作　細江　慎治(株式会社スーパースィープ)
アニメーション製作　金久保　哲也(株式会社バンダイナムコゲームス)
2Dグラフィックス　近藤敏信氏(有限会社スタジオ)
シナリオ　すずきこういち　(シナリオライター)

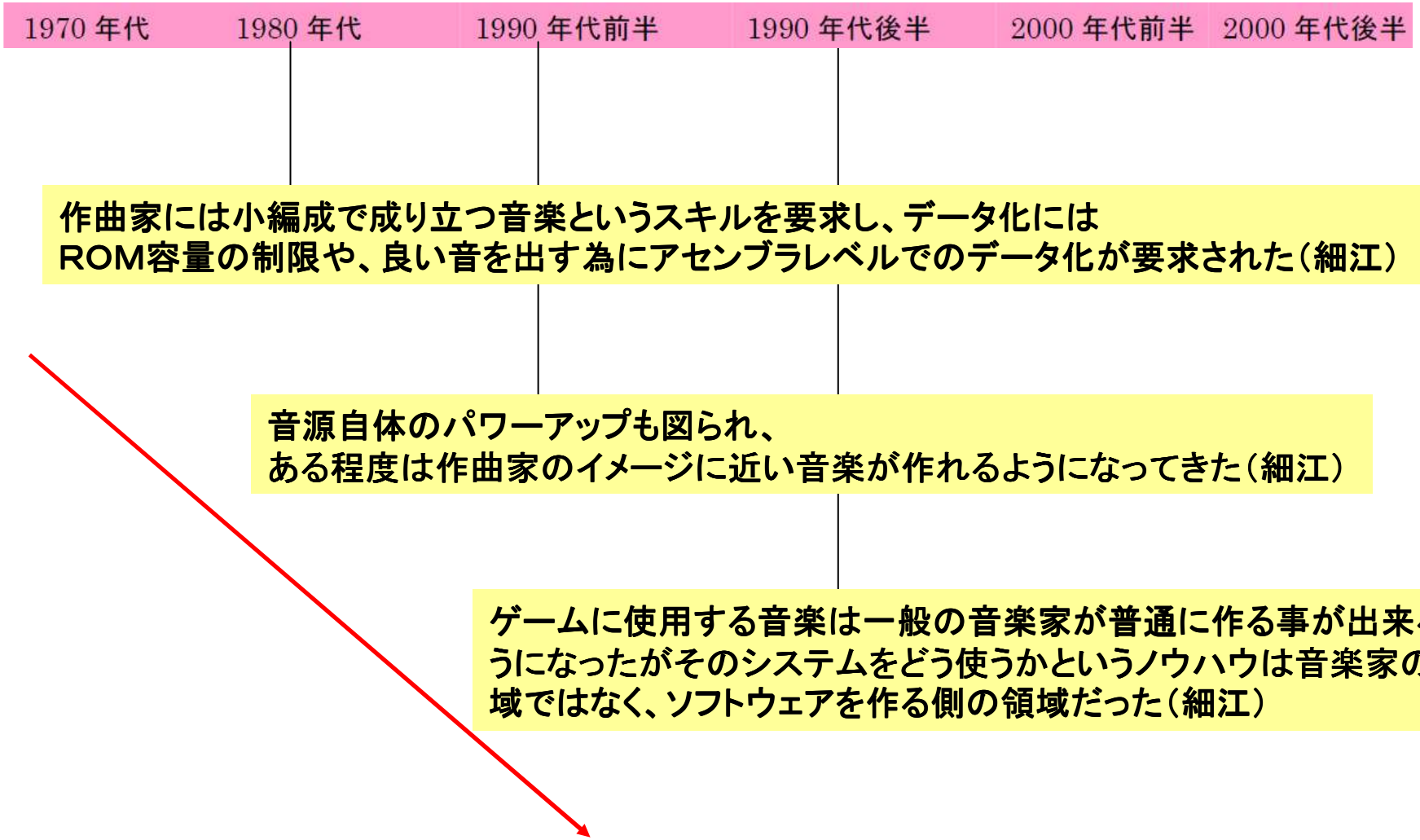
プログラミング　スクリプト　小久保　啓三(専門学校HAL東京)

アセンブラ　山口　誠　(株式会社グッド・フィール)
プログラミングAI　三宅　陽一郎(株式会社フロム・ソフトウェア)
プログラミング　グラフィックス描画　宮澤 篤　大久保 明(株式会社バンダイナムコゲームス)
プログラミング　物理・衝突判定　長谷川　晶一(電気通信大学)
3Dグラフィックス　シェーダー　今給黎隆　(株式会社バンダイナムコゲームス)
ゲーム物理シミュレーション　今給黎隆　(株式会社バンダイナムコゲームス)
タスクシステム技術　田村　祐樹　(株式会社ネバーランドカンパニー)　大野　功二(オープランニング)
黒須一雄(株式会社モバイル＆ゲームスタジオ)
ネットワーク通信　佐藤カフジ(フリーライター)

第二部　ゲーム開発技術の歴史概要

① データコンテンツの発展
② スクリプトの発展
③ プログラミング技術の発展

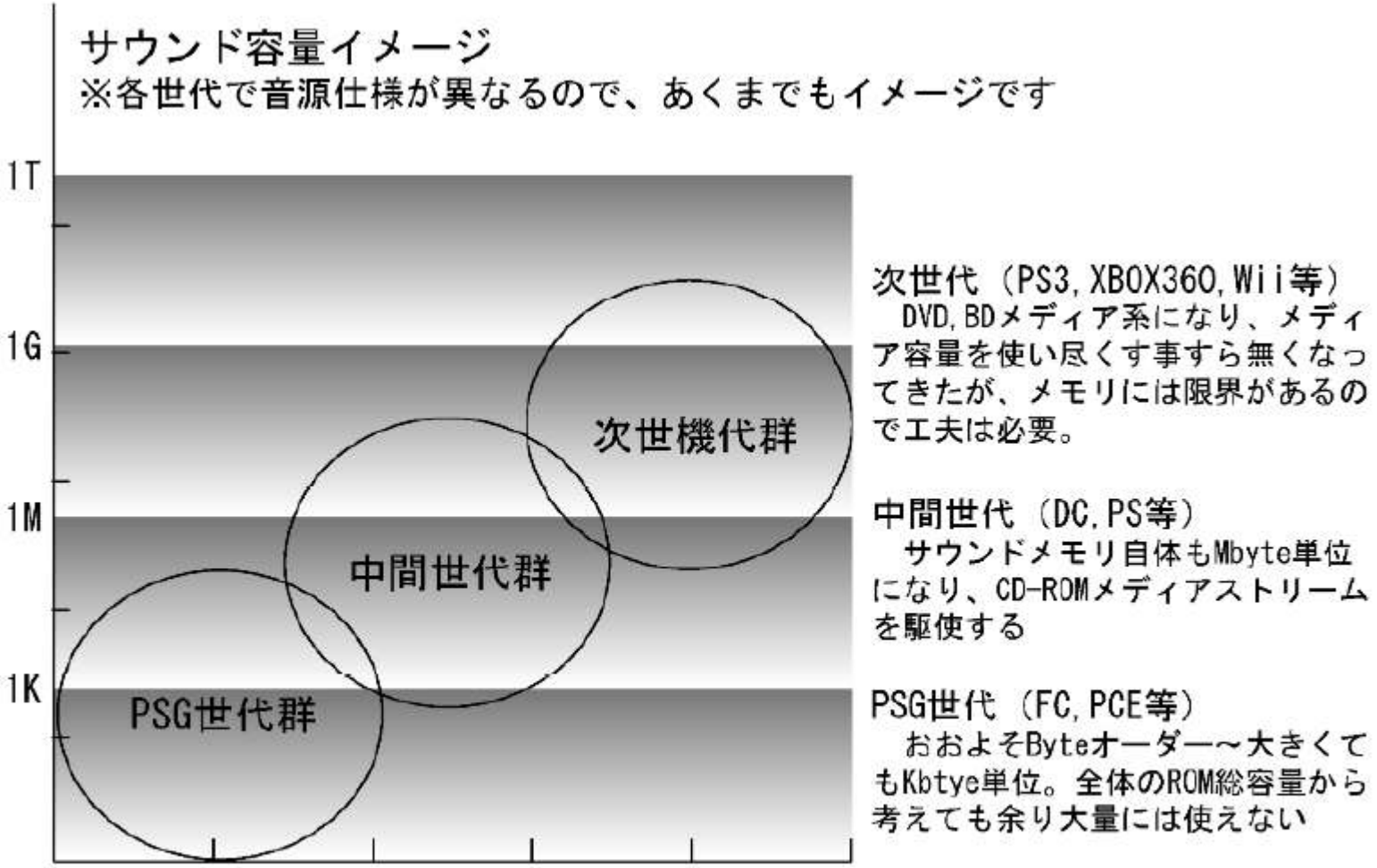
データコンテンツの発展（サウンド）



（2008年度報告書 「サウンド」 細江）

データコンテンツの発展（サウンド）

1970 年代	1980 年代	1990 年代前半	1990 年代後半	2000 年代前半	2000 年代後半



（2008年度報告書 「サウンド」 細江）

図 3.2-14　サウンド容量イメージ

データコンテンツの発展（アニメーション）

ポリゴンの表示はフラットシェーディングの単色で、表示できるポリゴン数も限られていた（大久保）

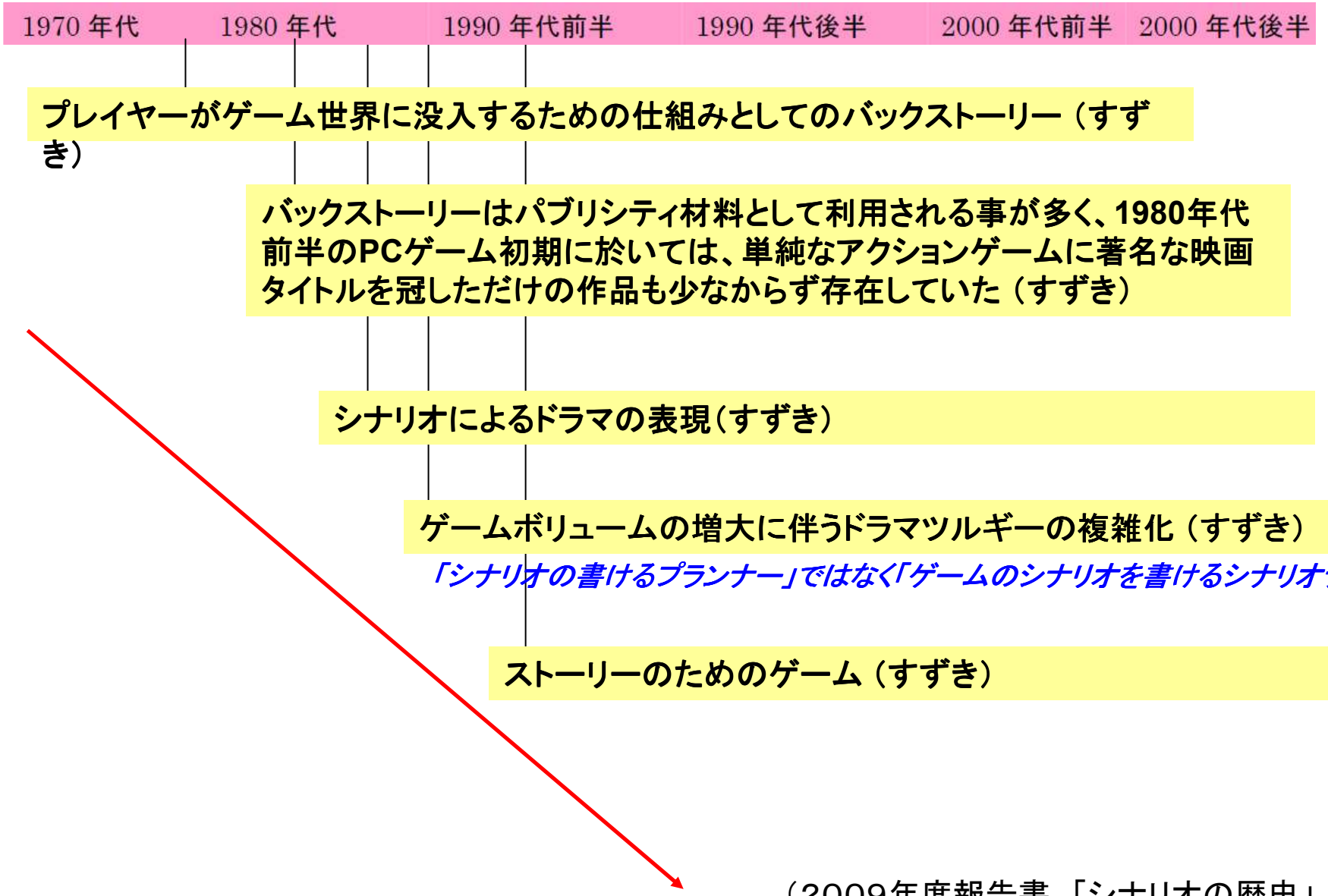
より多くのポリゴンが表示できるようになり、テクスチャマッピング機能が実装された事で、細かなグラフィック表現が出来るようになった（大久保）

家庭用市場と業務用市場の勢いは逆転
業務用ビデオゲームと比較して家庭用ビデオゲームは、
プレイ時間が長く、グラフィック、アニメーションなど大量のコンテンツが必要とされ、ゲーム開発の大規模化（大久保）

HDに対応した事によってグラフィックのクオリティを大幅に向上
アニメーション表現には、より細かな演技が求められるようになった。
2000年前半から引き続き、作成するコンテンツ量は増加傾向（大久保）

（2008年度報告書 「アニメーション」 大久保）

データコンテンツの発展（シナリオ）



（2009年度報告書 「シナリオの歴史」 すずき）

データコンテンツの発展（シナリオ）

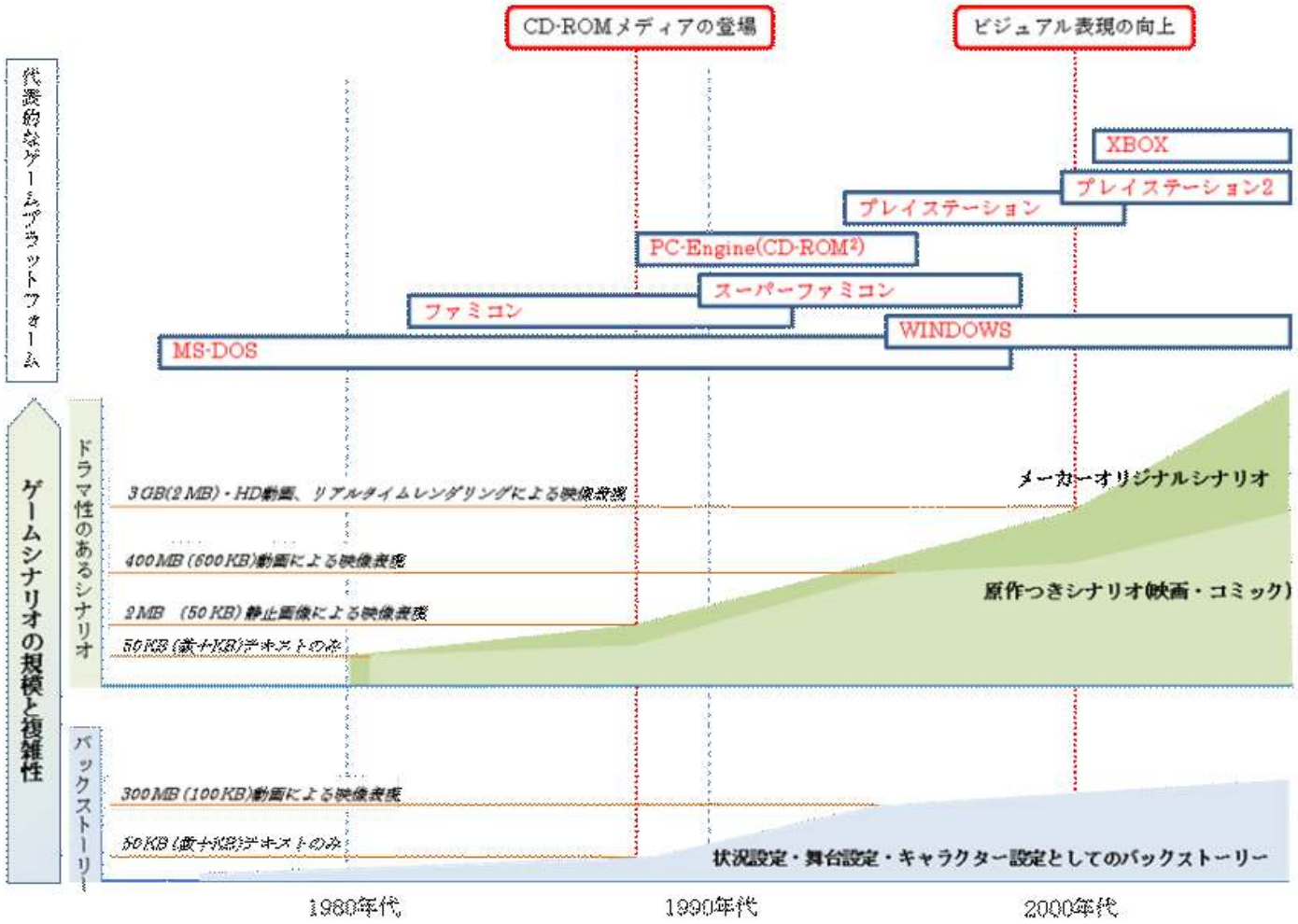


図 2-7-01 ゲーム中におけるゲームシナリオの規模と複雑性の変遷
容量表記は、「全体シナリオ量（画面に見えるテキスト量）」とした。

（2009年度報告書 「シナリオの歴史」 すずき）

第二部　ゲーム開発技術の歴史概要

① データコンテンツの発展
② スクリプトの発展
③ プログラミング技術の発展

スクリプト言語の発展

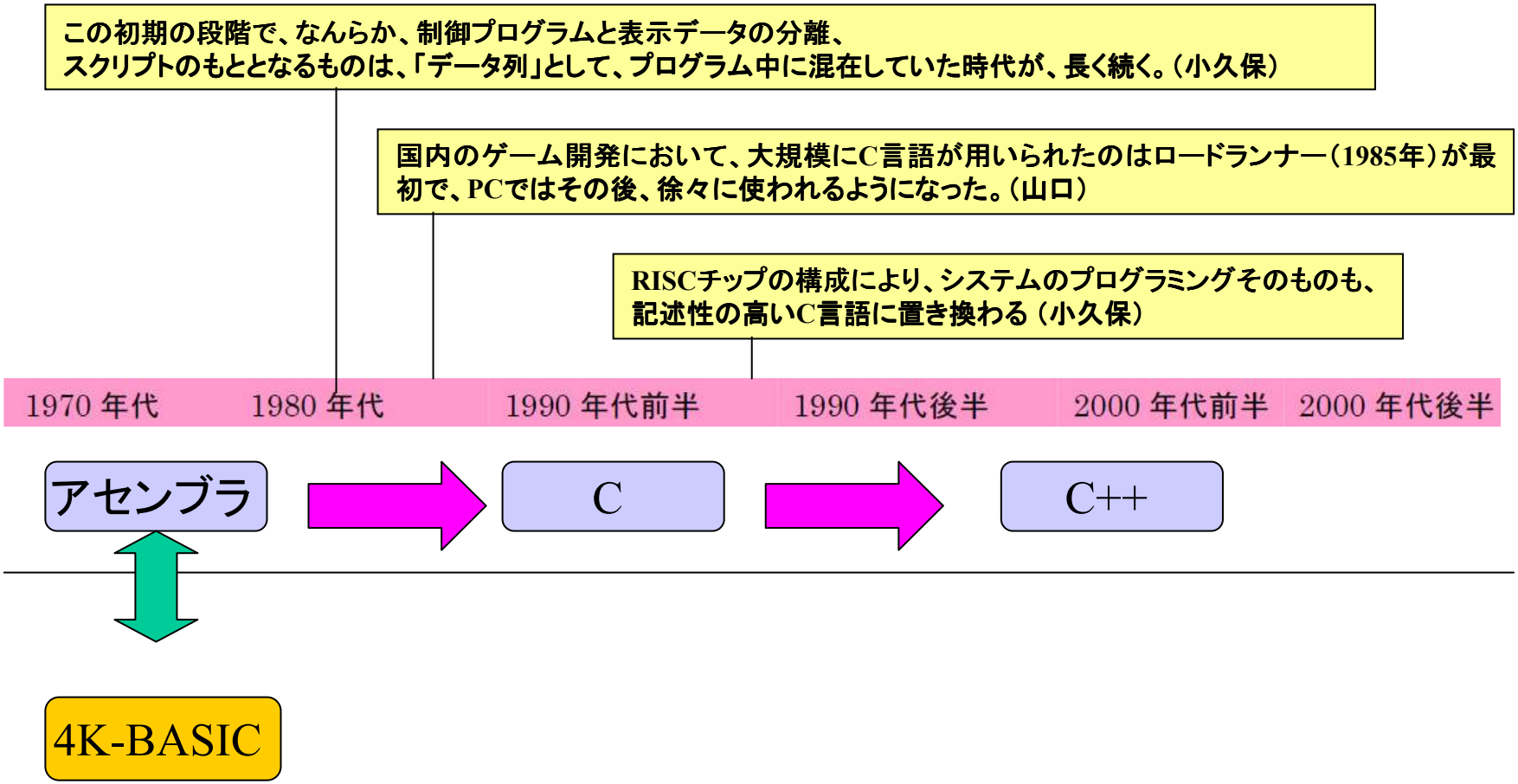
表 3.2-06　スクリプティングの歴史

（2008年度報告書 「スクリプティングの歴史」 大久保）

第二部　ゲーム開発技術の歴史概要

① データコンテンツの発展
② スクリプトの発展
③ プログラミング技術の発展

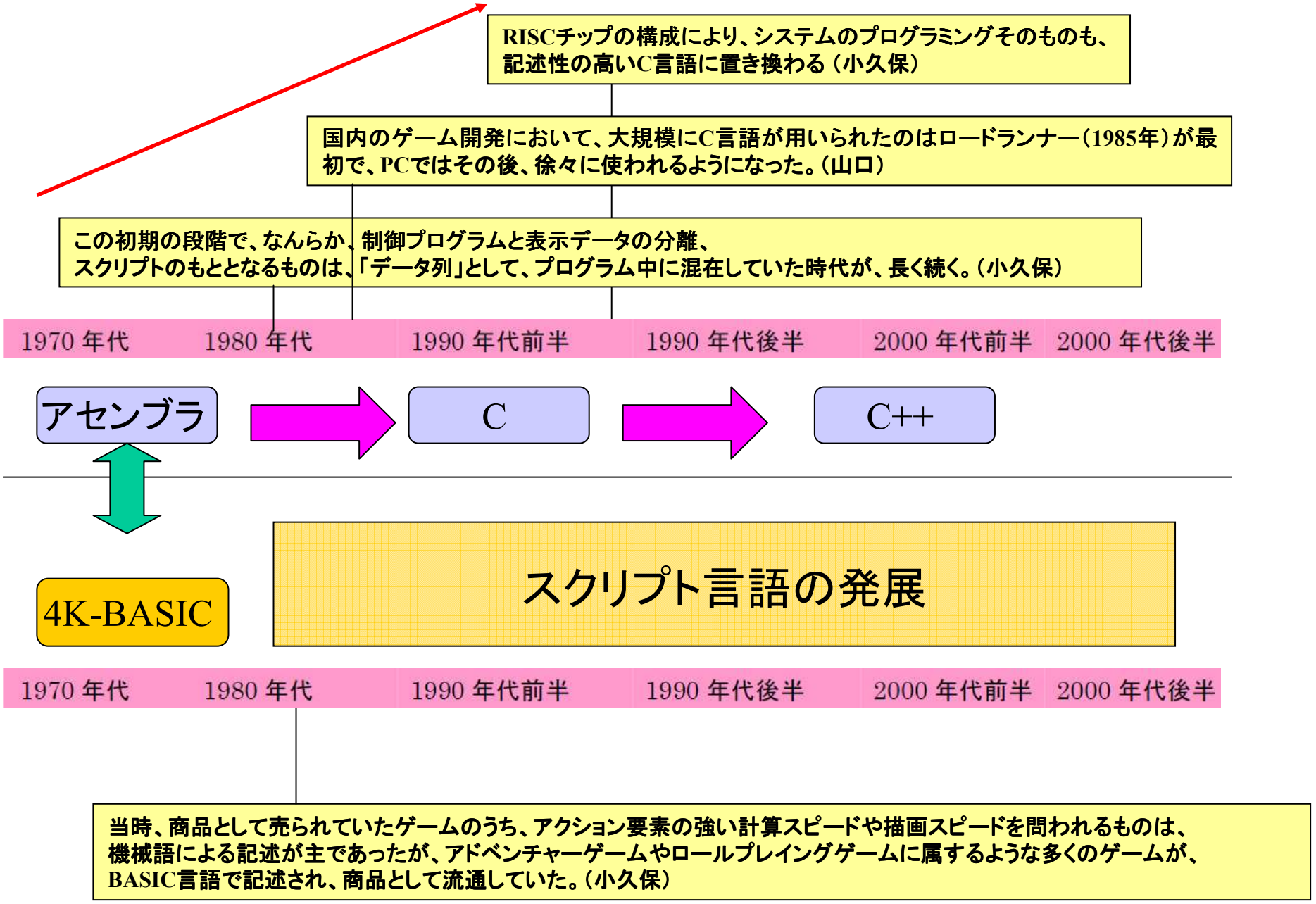
ゲーム・プログラミング言語の変遷



インタプリタ…遅い。遅くてもよいゲーム

当時、商品として売られていたゲームのうち、アクション要素の強い計算スピードや描画スピードを問われるものは、
機械語による記述が主であったが、アドベンチャーゲームやロールプレイングゲームに属するような多くのゲームが、
BASIC言語で記述され、商品として流通していた。（小久保）

ゲーム･プログラミング言語の変遷



ゲームデータの3つの要素による執筆3分野

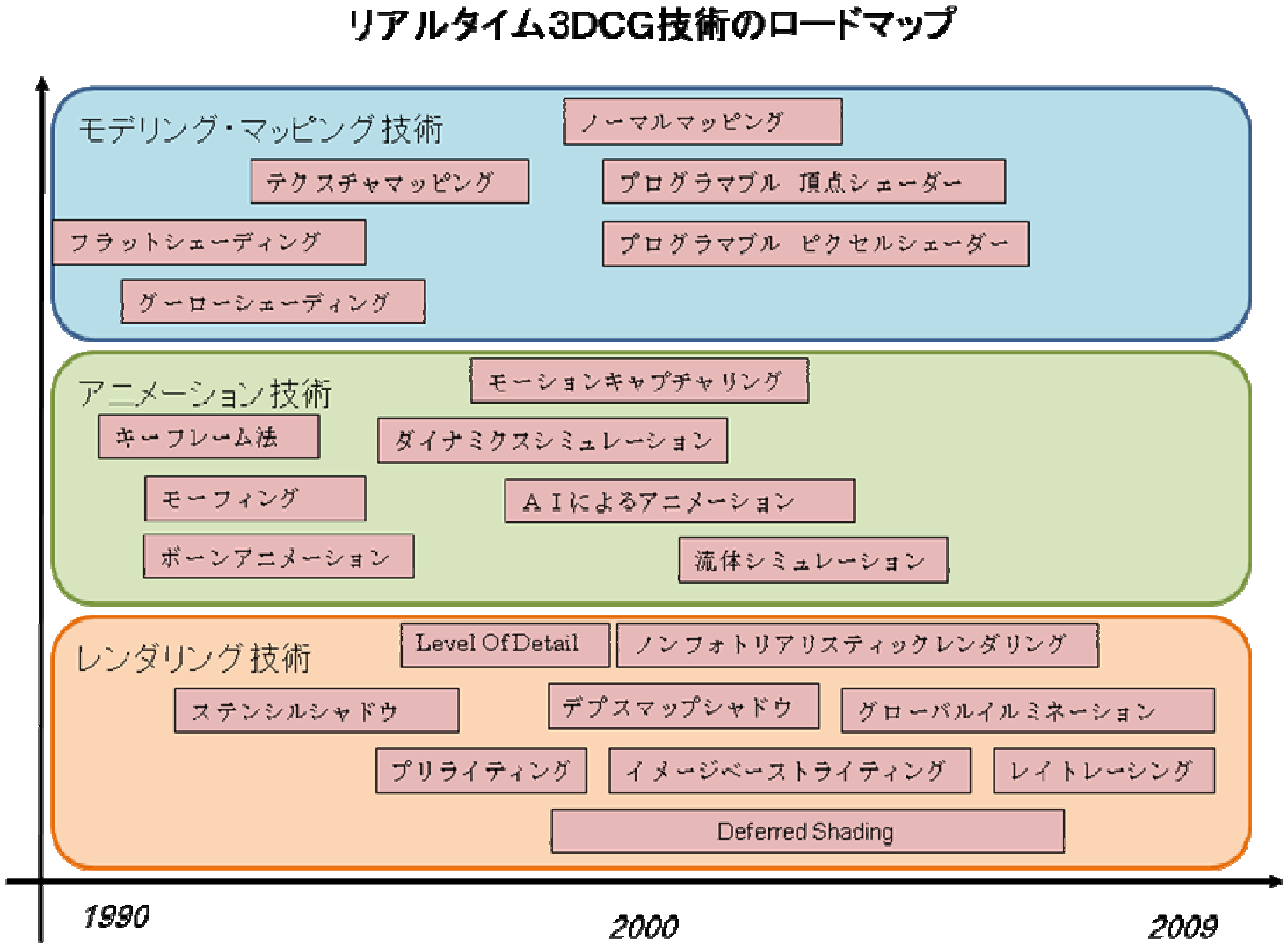
3D CG製作　川島 基展(東京工科大学 片柳研究所クリエイティブ・ラボ)
サウンド製作　細江　慎治(株式会社スーパースィープ)
アニメーション製作　金久保　哲也(株式会社バンダイナムコゲームス)
2Dグラフィックス　近藤敏信氏(有限会社スタジオ)
シナリオ　すずきこういち　(シナリオライター)

プログラミング　スクリプト　小久保　啓三(専門学校HAL東京)

アセンブラ　山口　誠　(株式会社グッド・フィール)
プログラミングAI　三宅　陽一郎(株式会社フロム・ソフトウェア)
プログラミング　グラフィックス描画　宮澤 篤　大久保 明(株式会社バンダイナムコゲームス)
プログラミング　物理・衝突判定　長谷川　晶一(電気通信大学)
3Dグラフィックス　シェーダー　今給黎隆　(株式会社バンダイナムコゲームス)
ゲーム物理シミュレーション　今給黎隆　(株式会社バンダイナムコゲームス)
タスクシステム技術　田村　祐樹　(株式会社ネバーランドカンパニー)　大野　功二(オープランニング)
黒須一雄(株式会社モバイル＆ゲームスタジオ)
ネットワーク通信　佐藤カフジ(フリーライター)

プログラミング技術の発展（CG）

1970 年代	1980 年代	1990 年代前半	1990 年代後半	2000 年代前半	2000 年代後半



（2008年度報告書 「3DCGの歴史」 川島）

プログラミング技術の発展（物理）

年表

現実の遊びを模倣するシミュレーション	
1958 年	『Tennis for Two』での簡易テニスシミュレーション
1979 年	『Flight Simulator』でのフライトシミュレーション
1981 年	『Raster Blaster』でのピンボールシミュレーション
1983 年	『3-D ゴルフシミュレーション』での 3D のゴルフシミュレーション
1988 年	『ハードドライビング』や『ウイニングラン』等のドライブシミュレーション

現実に近づけるシミュレーション	
慣性	
1958 年	『Tennis for Two』での重力加速度シミュレーション
1979 年	『Lunar Lander』での慣性シミュレーション
揺れ物	
1992 年	『インクレディブル・マシーン』での紐シミュレーション
1993 年	『バーチャファイター』での髪の毛シミュレーション
1994 年	『海腹川背』でのゴムロープシミュレーション
1994 年	『バーチャファイター2』でのクロスシミュレーション
流体	
1996 年	『ウエーブレース 64』での高さ場シミュレーション
2002 年	『DOUBLE-S.T.E.A.L.』でのピクセル単位のシミュレーション
2008 年	『AQUAFOREST Powered by OctaveEngine Casual』での 2 次元流体シミュレーション
剛体	
1998 年	『Jurassic Park: Trespasser』での剛体シミュレーション、ラグドール
2008 年	『Star Wars: The Force Unleashed』での破壊シミュレーション

（2009年度報告書 「ゲーム物理」 今給黎）

プログラミング技術の発展（シェーダー）

- 年表

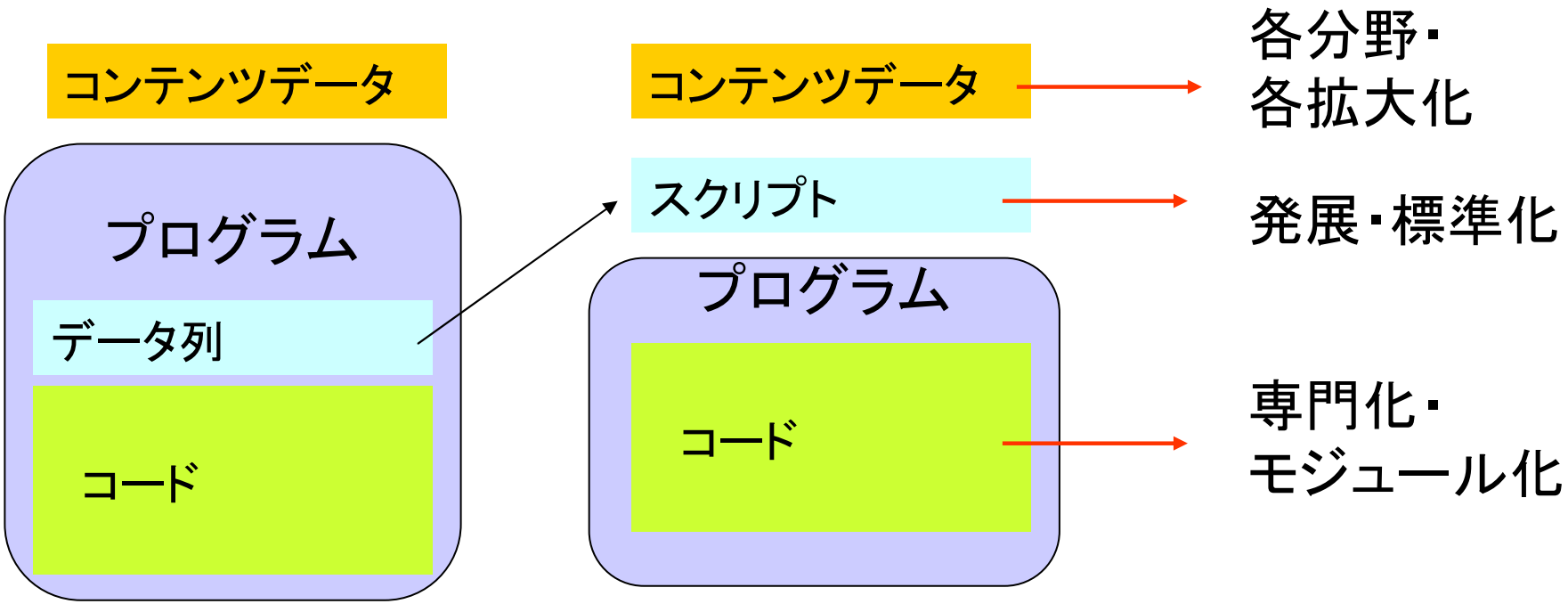
年代	名称等	種類
1984	Shade trees	論文
1986	PhotoRealistic RenderMan	レンダラー
1996	Voodoo Graphics, PowerVR	ビデオカード
1998	RIVA TNT	ビデオカード
1998	R4 -RIDGE RACER TYPE 4-(ベイク)	タイトル
1998	Half-Life(ハーフランバート)	タイトル
1999	RIVA TNT2	ビデオカード
2000	DirectX 8.0	API
2001	GeForce3	ビデオカード
2002	DOUBLE-S.T.E.A.L(グレア)	タイトル
2002	DirectX 9.0	API
2004	ハーフライフ 2(法線マッピング、トーンマッピング)	タイトル
2005	ワンダと巨像（トーンマッピング）	タイトル
2006	ヴァルキリープロファイル 2（半球ライティング）	タイトル
2006	バーチャファイター5（リムライトシェーダ）	タイトル
2006	MT フレームワーク（モーションブラー）	ゲームエンジン
2006	DirectX 10.0	API
2007	Halo3(球面調和関数展開した金属反射など)	タイトル
2007	Killzone2(遅延レンダリング)	タイトル
2007	Cyrsis(SSAO)	タイトル
2008	ソニックワールドアドベンチャー(光伝搬ボリューム)	タイトル
2009	Halo WARS(細分割曲面)	タイトル
2009	DirectX 11	API

（2009年度報告書 「シャーダーの歴史」 今給黎）

この初期の段階で、なんらか、制御プログラムと表示データの分離、
物語の制御コードもデータとして分離するようなプログラミングが行われていた。

スクリプトのもととなるものは、「データ列」として、プログラム中に混在していた時代が、長く続く。
比較的小規模なゲームプログラムであれば、これは今も同じだろう。（小久保）

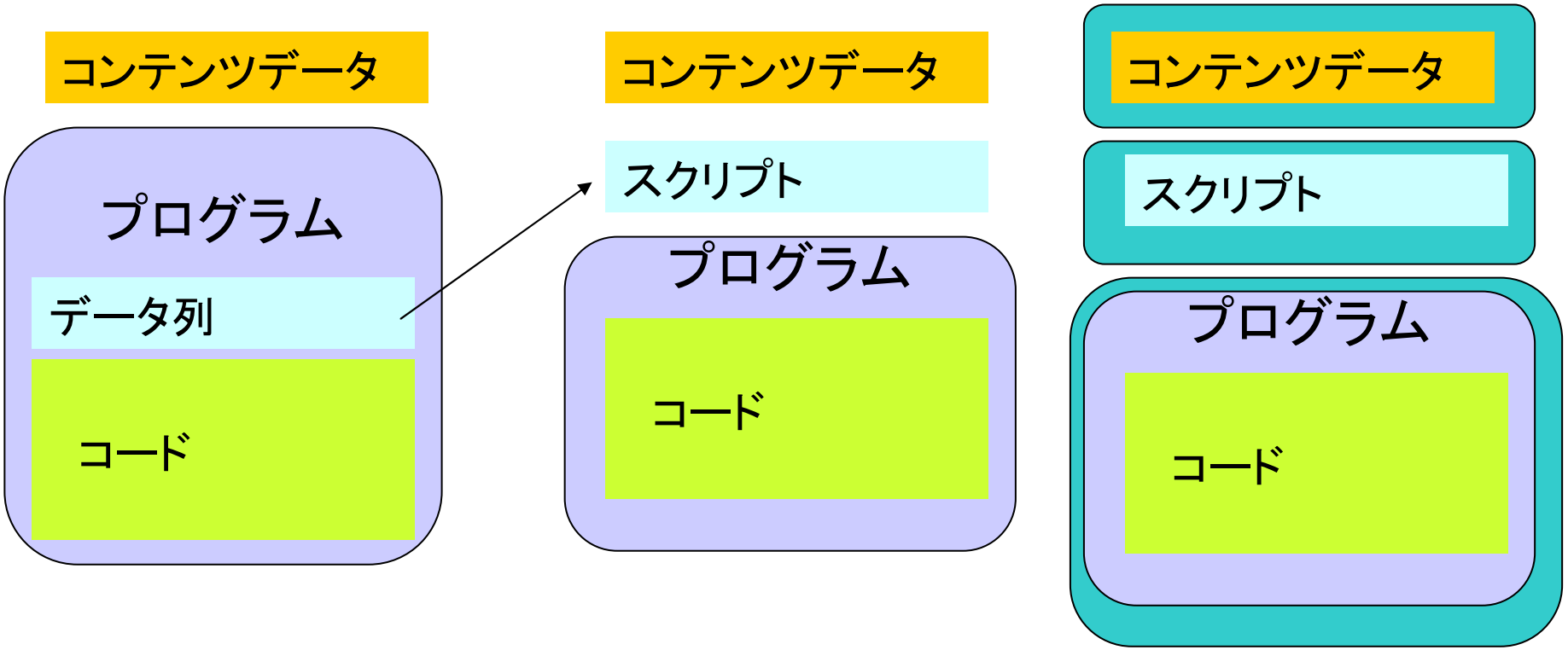
1970 年代	1980 年代	1990 年代前半	1990 年代後半	2000 年代前半	2000 年代後半



この初期の段階で、なんらか、制御プログラムと表示データの分離、
物語の制御コードもデータとして分離するようなプログラミングが行われていた。

スクリプトのもととなるものは、「データ列」として、プログラム中に混在していた時代が、長く続く。
比較的小規模なゲームプログラムであれば、これは今も同じだろう。（小久保）

1970 年代	1980 年代	1990 年代前半	1990 年代後半	2000 年代前半	2000 年代後半



コンテンツをミドルウェアでラップする

ミドルウェアの役割

(1) プラットフォームの抽象化
(2) マルチプラットフォーム化
(3) 技術要素の高度化
(4) インハウスのゲーム技術開発との関係

（2009年度報告書 「ミドルウェア」 佐藤）

ミドルウェアの発展

x .3.1　ゲームエンジン

表-x.3-01　代表的なゲームエンジン

名称	提供元	対応機種	特徴	主なジャンル
Unreal Engine 3	EPIC Games	Windows / Xbox 360 / PlayStation 3 /	高品質グラフィックス / 高度なレベルエディタ / スクリプトエディタ	FPS 3D アクション
Source	Valve	Windows / Xbox 360 /	レベルエディタ等周辺ツールがオープン/ 拡張性・柔軟性が極めて高い	FPS 3D アクション
CryEngine 2	Crytek	Windows / Xbox 360 / PlayStation 3	高品質グラフィックス / ゲームのランタイム編集 / 独自物理エンジン	FPS

（2009年度報告書 「ミドルウェア」 佐藤）

ミドルウェアの発展

x .3.2　開発フレームワーク

表-x.3-02　代表的な開発フレームワーク

名称	提供元	対応機種	特徴	主なジャンル
Alchemy	Activision / シリコンスタジオ	Windows / Xbox 360 / PS2 / PS3 / PSP Wii	マルチプラットフォーム開発の高い実績 / ライセンス費が安価で小規模開発にも向く	3D ゲーム
Gamebryo	Emergent Technologies	Windows Xbox 360 PS2 / PS3 / Wii	高い柔軟性 / レベルエディタ / ランタイム編集 / PhysX ビルトイン	3D ゲーム
Unity	Technologies	Windows / Mac Wii / iPhone	Web ブラウザで動作するモジュールを出力可 / 統合開発環境 / 安価	ブラウザゲーム カジュアルゲーム モバイルゲーム
Torque Game Engine	GarageGames	Windows / Mac / Xbox 360 / Wii / iPhone	Web ブラウザで動作するモジュールを出力可 / 極めて安価	ブラウザゲーム カジュアルゲーム モバイルゲーム
BigWorld	BigWorld Technology	Windows	MMOG 開発・運営のための完全なテクノロジースイート	MMOG

（2009年度報告書 「ミドルウェア」 佐藤）

ミドルウェアの発展

表-x.3-02　代表的な単機能ミドルウェア

名称	提供元	機能	特徴	主なジャンル
PhysX	NVIDIA	物理処理	NVIDIA CUDA 対応 / 各プラットフォーム最適化 / 無料版あり	3D アクション 各種シミュレーター
Havok	Intel	物理処理	極めて豊富な採用実績 / 姉妹 AI エンジン、アニメーションエンジンと協調	3D アクション 各種シミュレーター
Kynapse	Autodesk	AI(経路探索)	経路探索を高速に処理 / 環境の変化にリアルタイム適応	3D アクション ビジュアルシミュレーション
HumanIK	Autodesk	アニメーション	ボーンアニメーションのリターゲッティング / UE3 に統合可能	3D ゲーム
BINK Video	RAD Game Tools	動画コーデック	高速 / 高画質 / 低ビットレート / 無数の採用実績	ジャンル問わず
YEBIS	シリコンスタジオ	リアルタイム・ポストエフェクト	プログラマなしにグラフィックス品質を調整 / Alchemy との親和性高	3D ゲーム
	コミュニティー		サードパーティアンド	

（2009年度報告書「ミドルウェア」 佐藤）

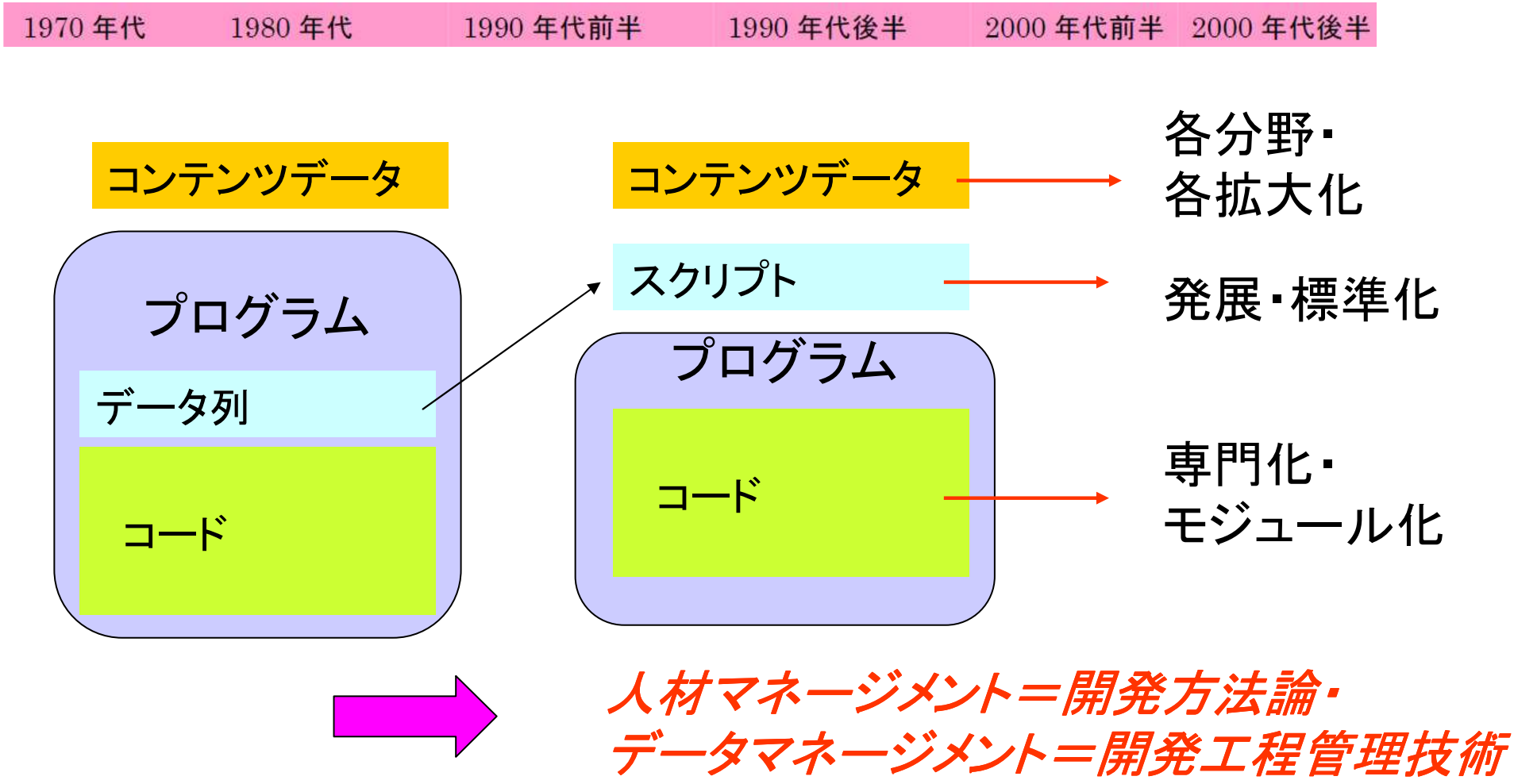
ミドルウェアの発展

VCE	エンジン	ネットワーク	型ネットワークゲームのフレームワーク	MMORPG
SpeedTree	IDV	植生エンジン	植生自動生成/ 手動編集 / ランタイム表示エンジン / 高品質な映像表現	3D ゲーム
ScaleForm GFx	ScaleForm	UI エンジン	Adobe Flash で作成した UI 要素をゲームにインポート / 動画再生支援	3D ゲーム 2D ゲーム
ファイルマジック PRO	CRI ミドルウェア	ファイルシステム	透過的にゲームデータの圧縮・復号を支援 / ファイルシステム最適化	ジャンル問わず
Dolby Axon	Dolby	ボイスチャット	3D 空間の角度、距離、環境要素を考慮してボイスチャット音声を再現	3D マルチプレーヤーゲーム

（2009年度報告書 「ミドルウェア」 佐藤）

この初期の段階で、なんらか、制御プログラムと表示データの分離、
物語の制御コードもデータとして分離するようなプログラミングが行われていた。

スクリプトのもととなるものは、「データ列」として、プログラム中に混在していた時代が、長く続く。
比較的小規模なゲームプログラムであれば、これは今も同じだろう。（小久保）



開発方法論　長久　勝　（ハイパーコンテンツ(株)/早稲田大学メディアネットワークセンター）
開発工程管理　田村　祐樹　（株式会社ネバーランドカンパニー）

第二部　ゲーム開発技術の歴史概要

① データコンテンツの発展
② スクリプトの発展
③ プログラミング技術の発展

④ 開発方法論
⑤ 開発工程管理技術

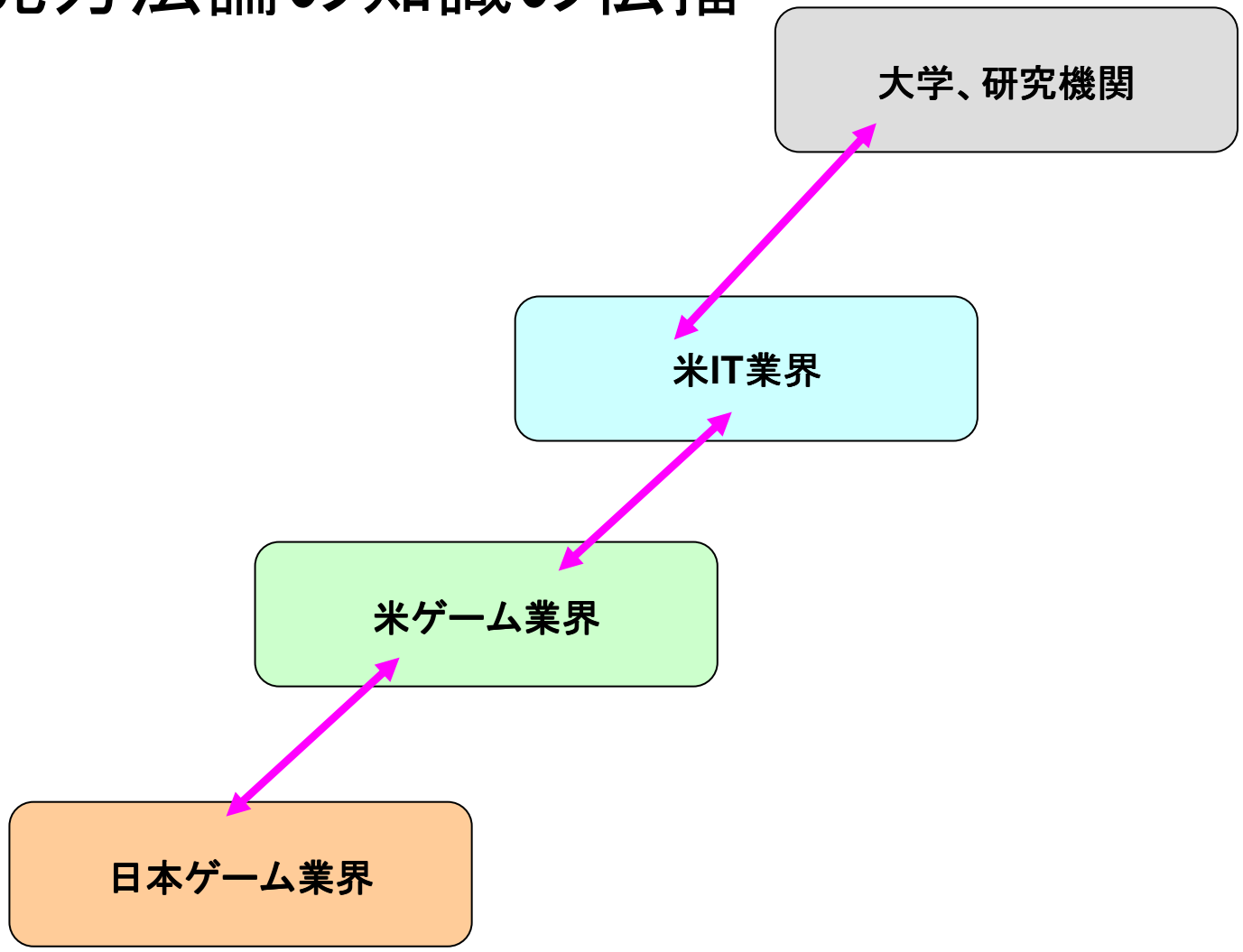
開発方法論の発展

表 3.2-07　関連年表

年	出来事、論文、刊行物など
1939	Walter A. Shewhart, W. Edwards Deming "Statistical method from the viewpoint of quality control"
1950	Deming が日本に PDC(S)A を紹介
1970	Winston W. Royce "Managing the development of large software systems"
1985	DOD-STD-2167
1986	野中 郁次郎, 竹内 弘高 "The New New Product Development Game"
1988	Barry Boehm "A Spiral Model of Software Development and Enhancement"
1990	James Womack, Daniel Jones, Daniel Roos "The Machine That Changed the World"
1993	Jeff Sutherland、John Scumniotales、Jeff McKenna、Ken Schwaber らが Scrum を始める
1996	Kent Beck が XP(Extreme Programming)を考案
1997	Valve 社が Half-Life の開発に Cabal を適用
2001	アジャイル宣言

（2008年度報告書「開発方法論」長久）

開発方法論の知識の伝播

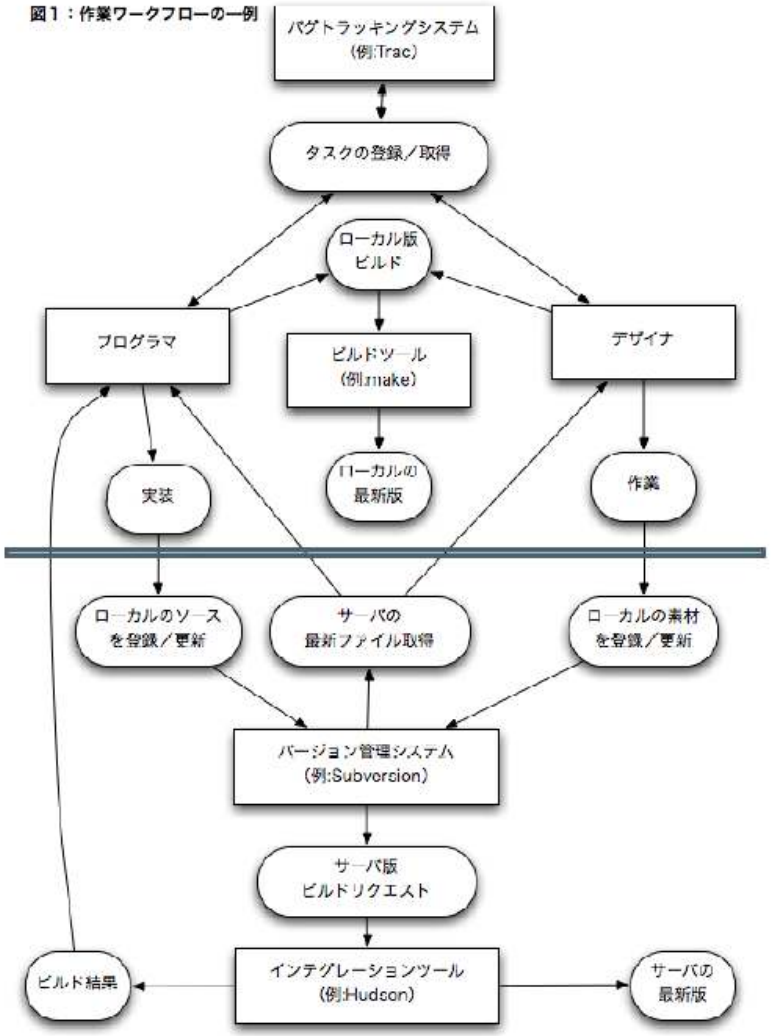


第二部　ゲーム開発技術の歴史概要

① データコンテンツの発展
② スクリプトの発展
③ プログラミング技術の発展

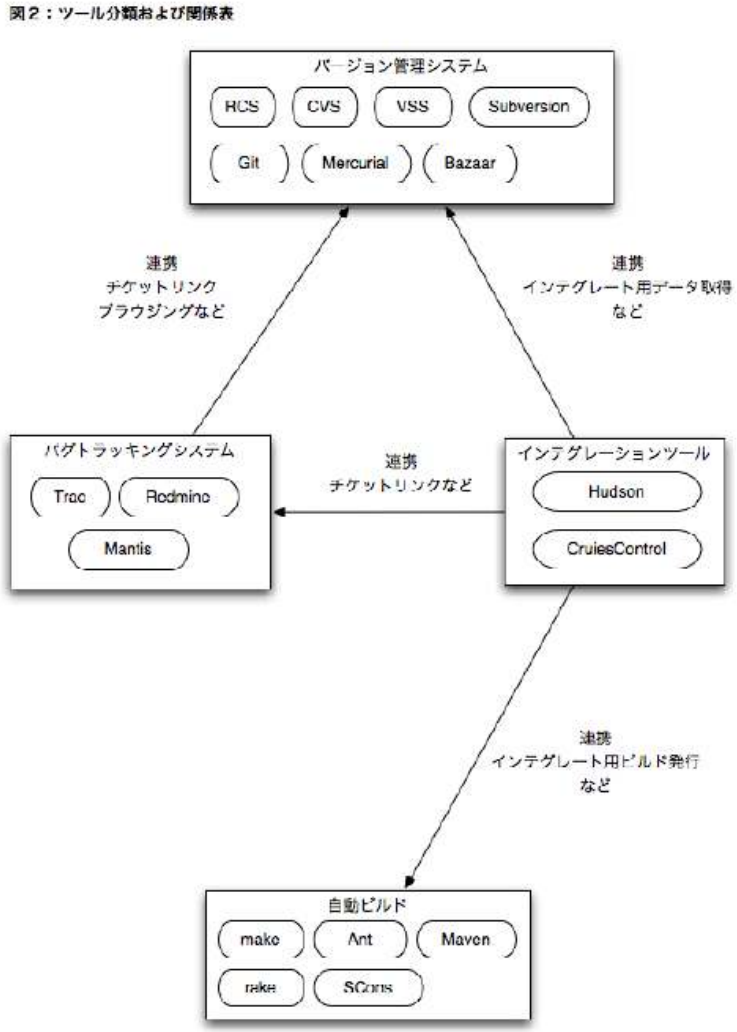
④ 開発方法論
⑤ 開発工程管理技術

開発工程管理技術の発展



（2009年度報告書 「開発方法論」 田村）

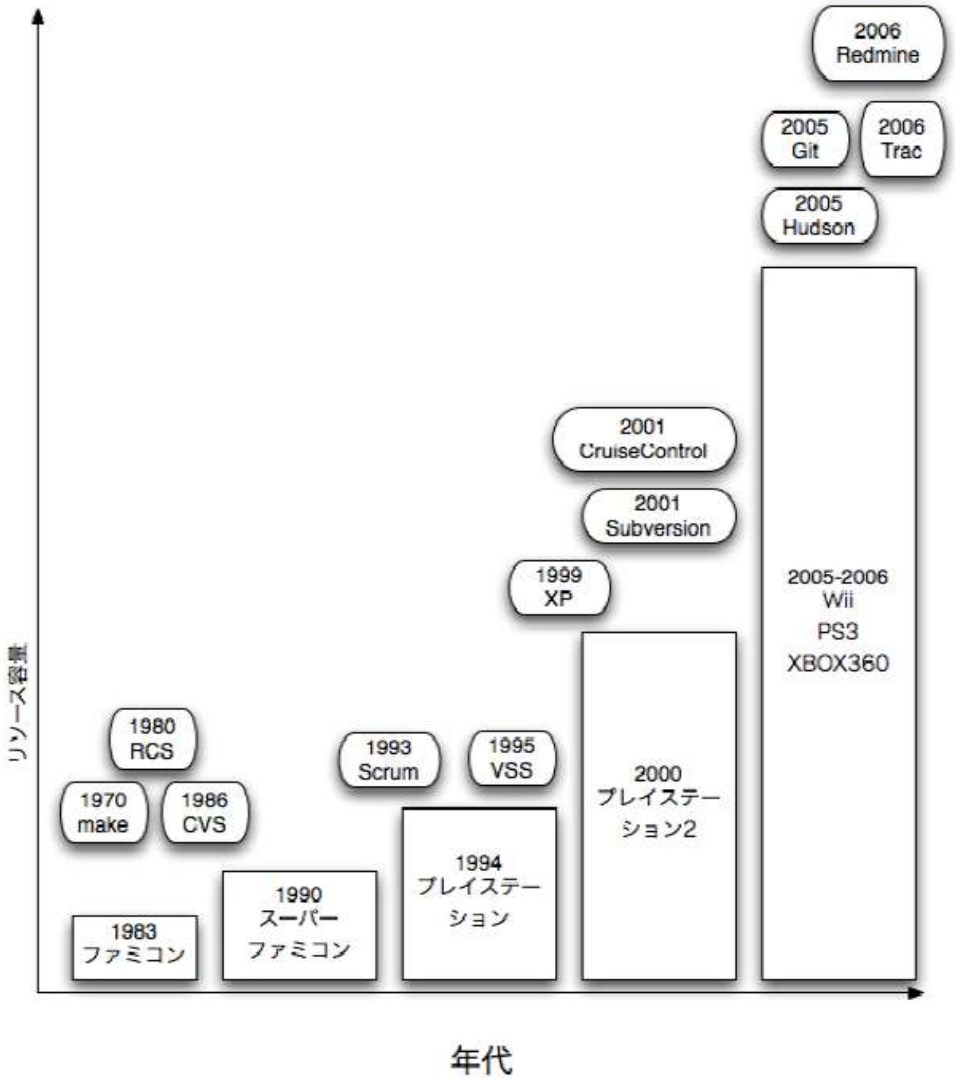
開発工程管理技術の発展



（2009年度報告書「開発方法論」田村）

開発工程管理技術の発展

図３：開発ツール年表



ハードウェア登場におけるリソース容量増加と
同時期に登場したツール

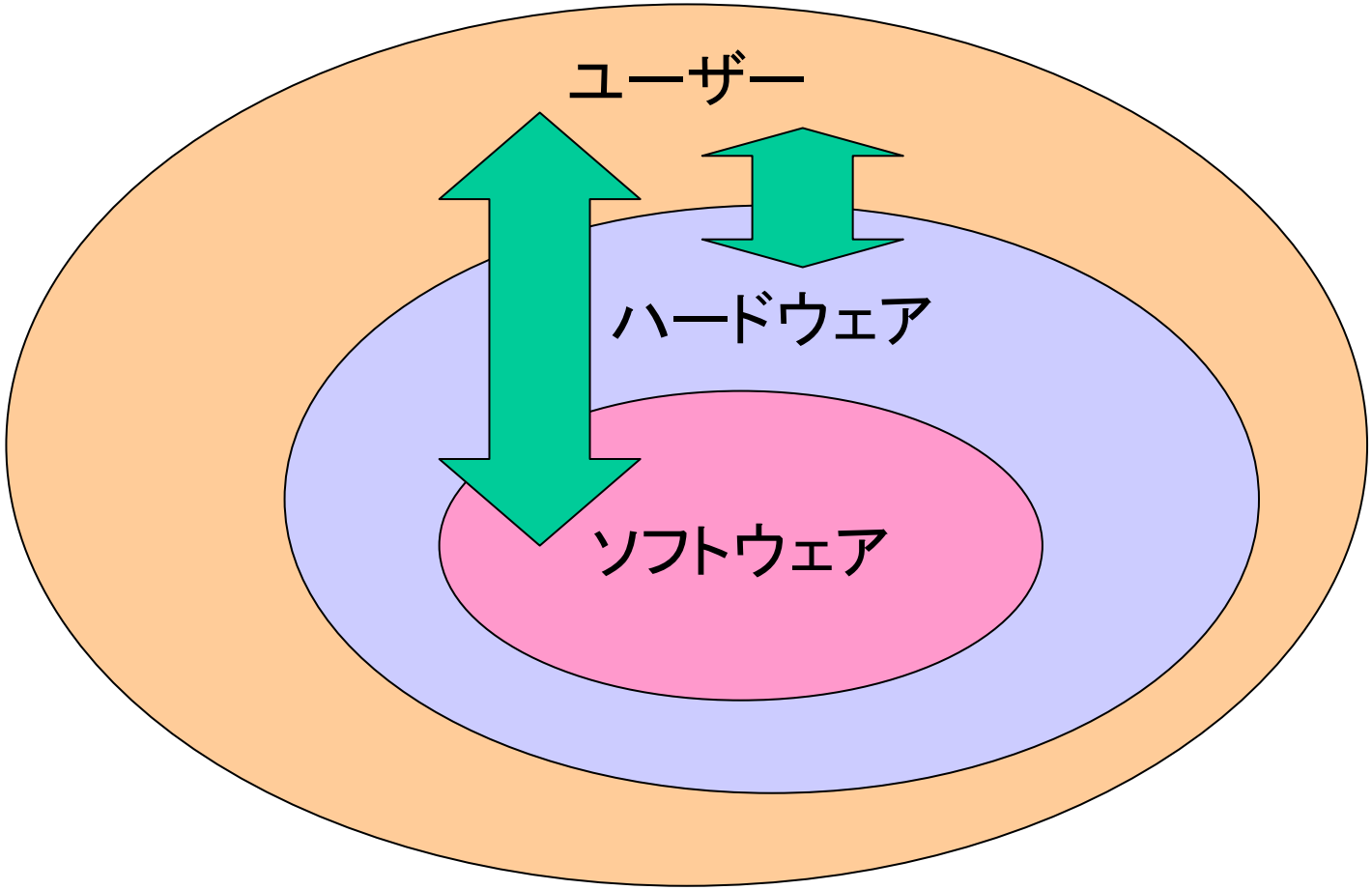
（2009年度報告書 「開発方法論」 田村）

第二部　ゲーム開発技術の歴史概要

① データコンテンツの発展
② スクリプトの発展
③ プログラミング技術の発展

④ 開発方法論
⑤ 開発工程管理技術

より大きくゲームを捉える…



ユーザーインターフェイス（ハードウェア）　大神佳人氏（株式会社HORI）

ユーザーインターフェイス（ソフトウェア）　サイトウアキヒロ氏（立命館大学）

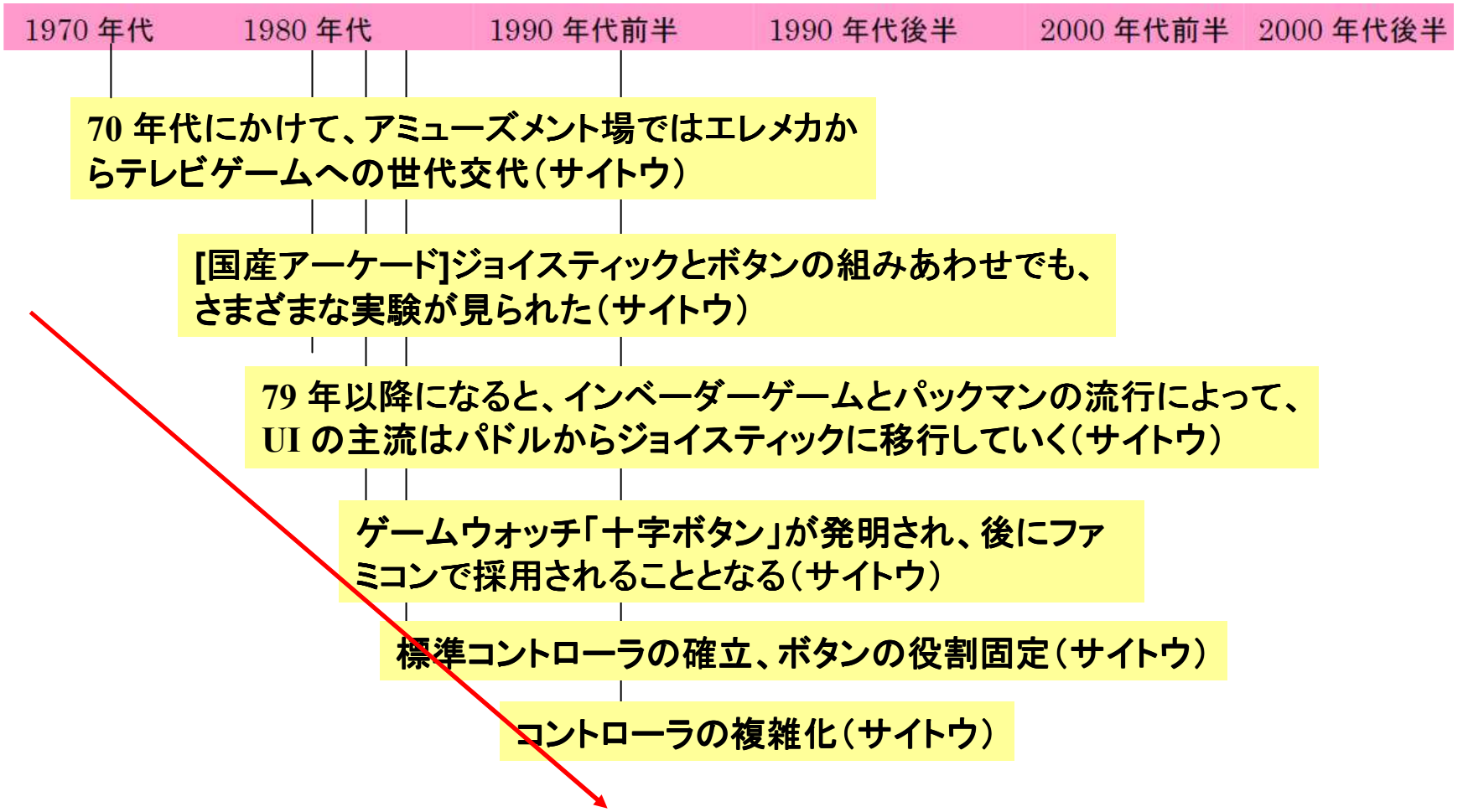
第二部　ゲーム開発技術の歴史概要

① データコンテンツの発展
② スクリプトの発展
③ プログラミング技術の発展

④ 開発方法論
⑤ 開発工程管理技術

⑥ ユーザーインターフェース

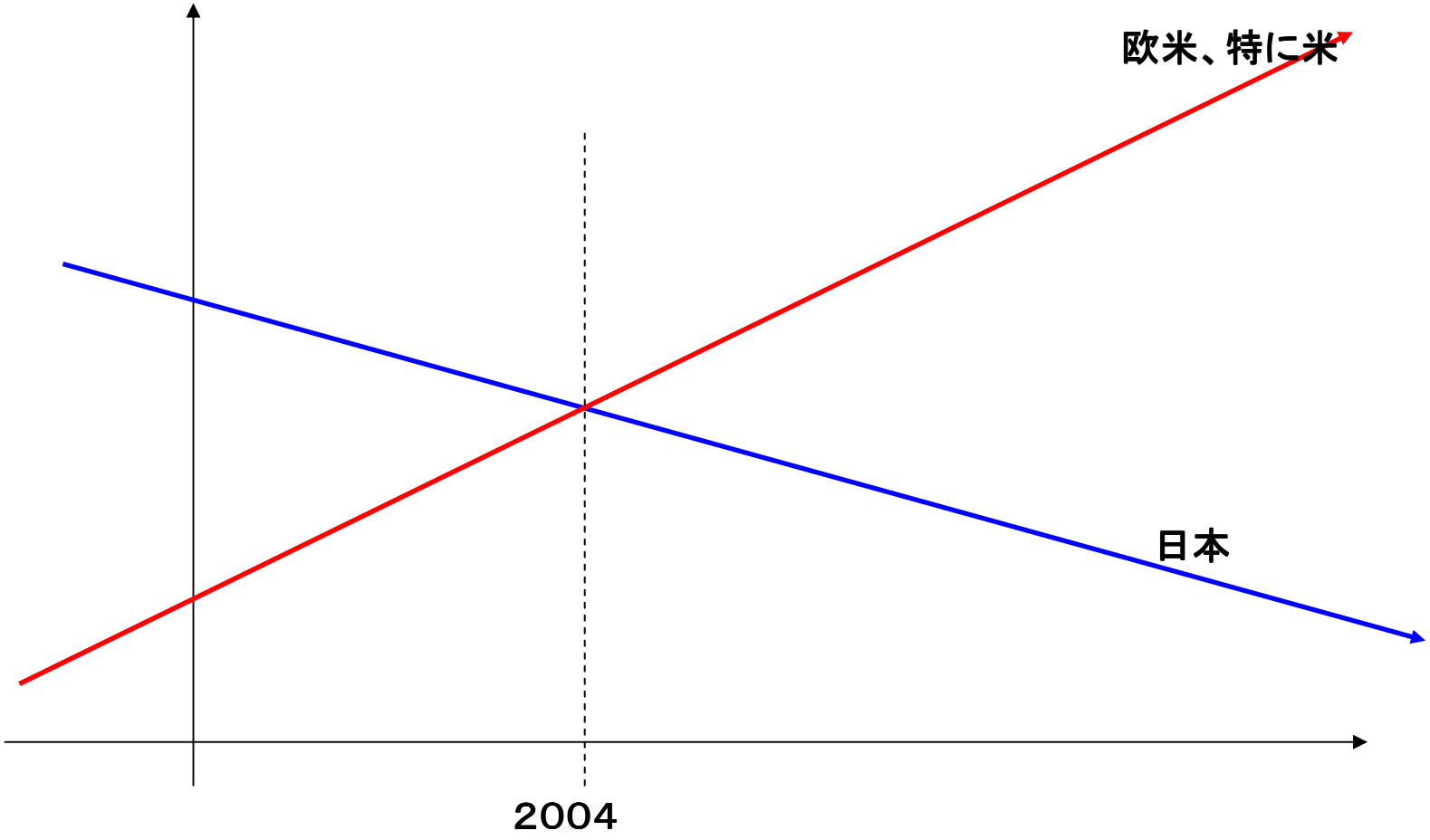
ユーザーインターフェースの発展



（2009年度報告書 「ユーザーインターフェースの歴史」 サイトウ）

第三部　まとめ

この10年のゲーム開発技術の変化のイメージ



技術書籍、技術論文の圧倒的な差

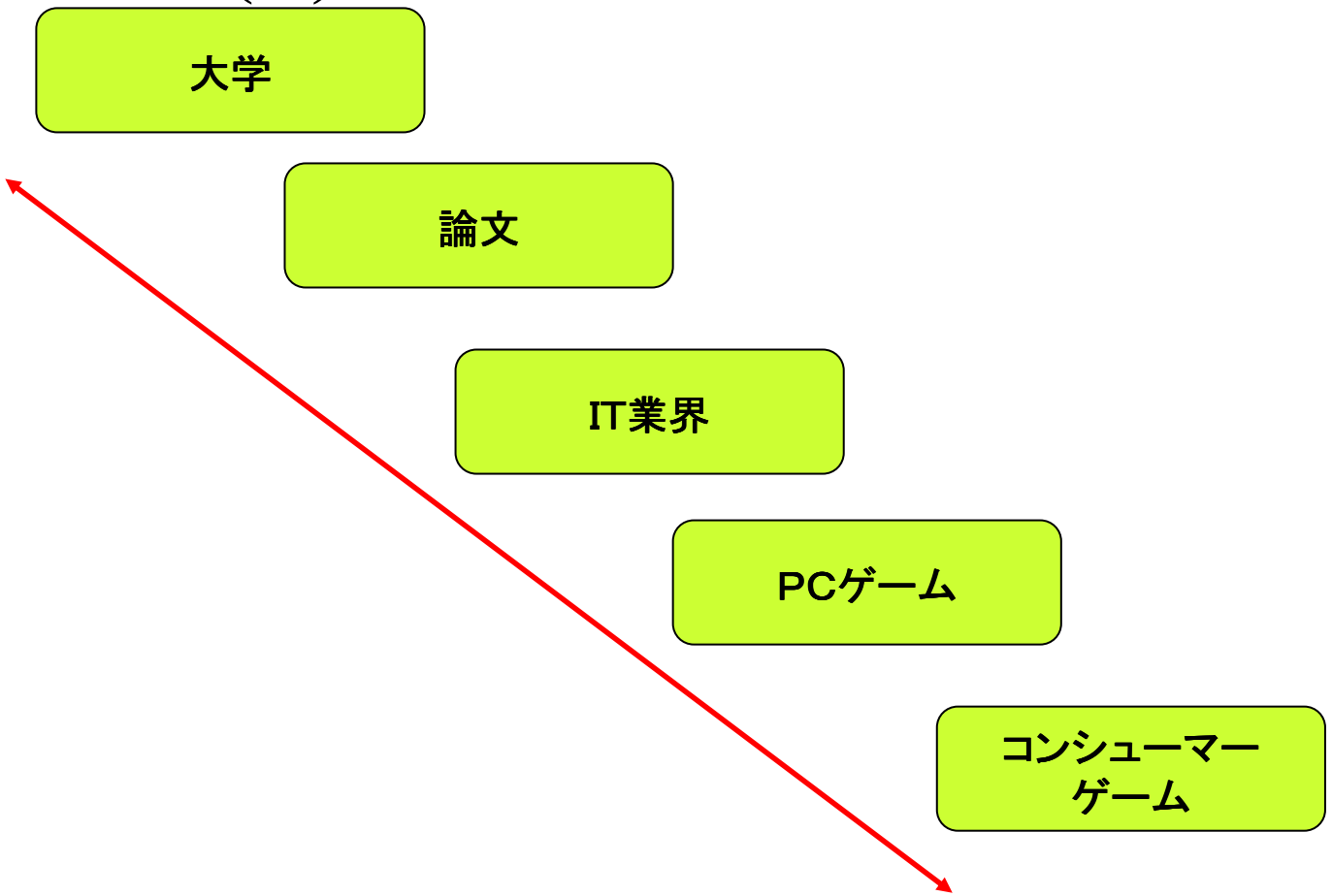
気づいたこと

殆どの技術情報は欧米から来ている

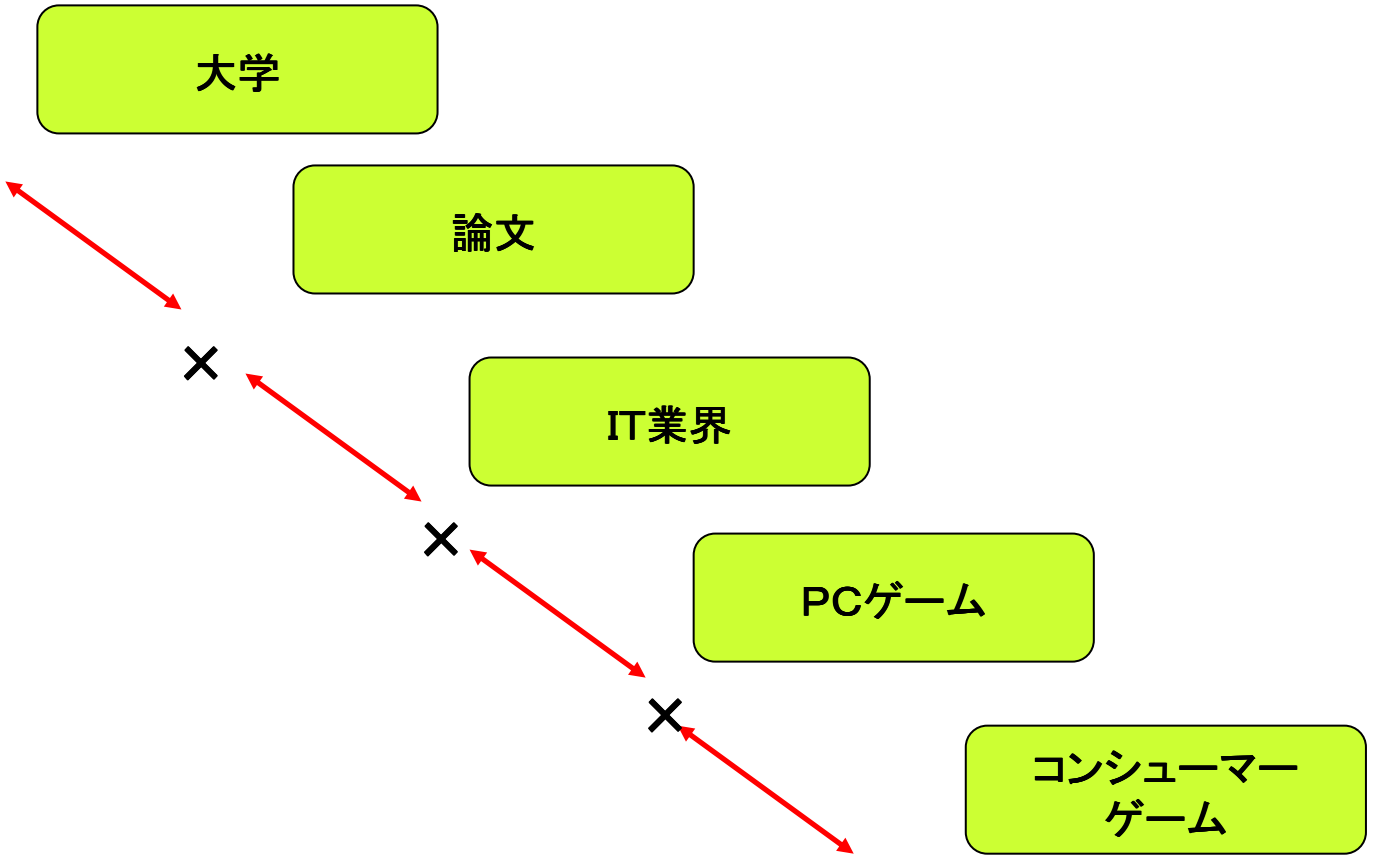
日本から生み出した技術の潮流はない

なぜ？

(1) 米における技術導入経路



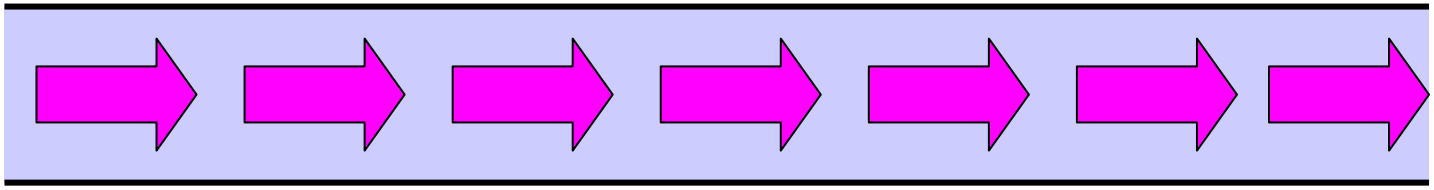
日本における技術導入経路



補給なし、援助なし、厳しい．．．

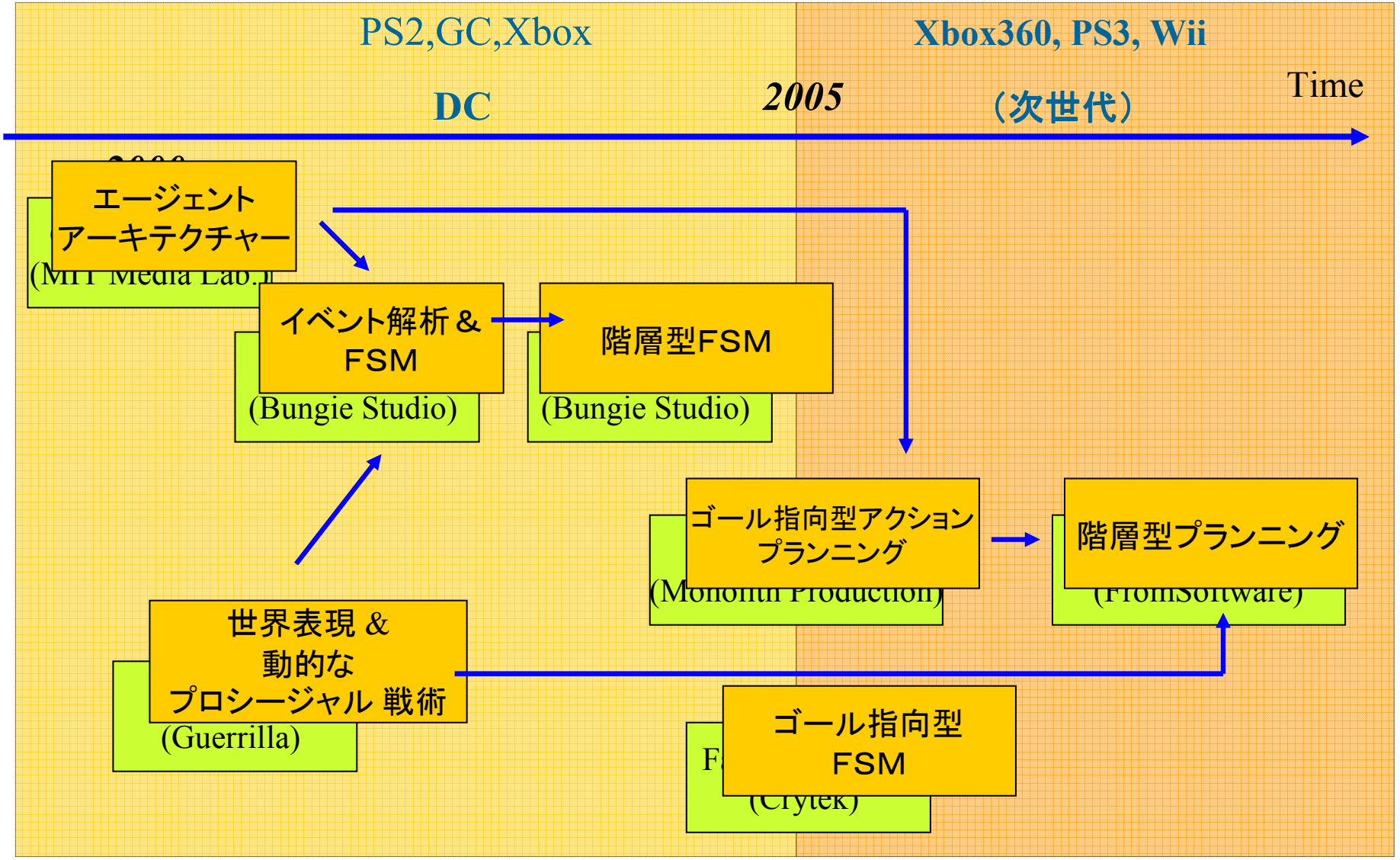
(2) 技術進化トンネル（AIの場合）

米のゲームAIは、主にFPSを苗床にして進化した。



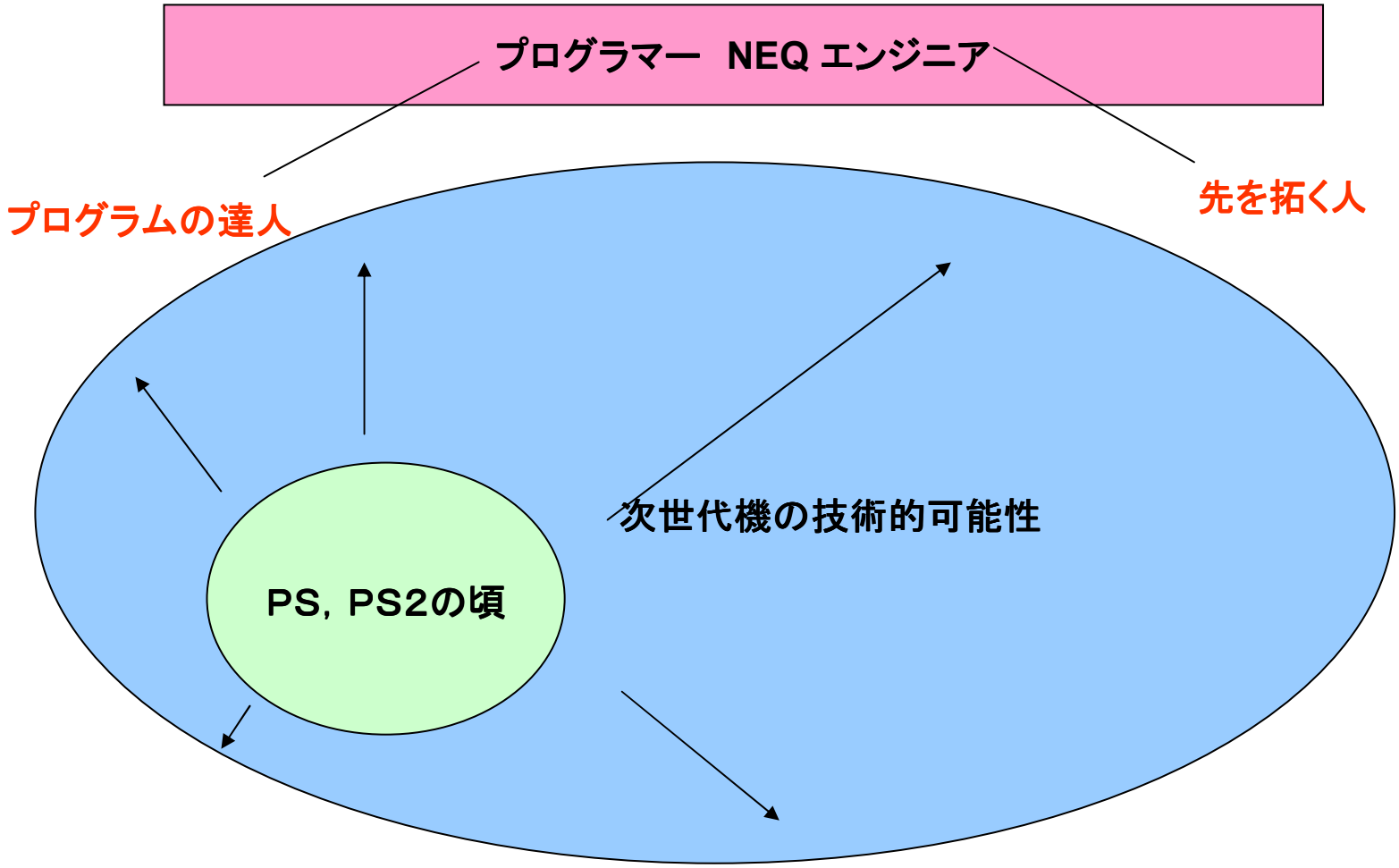
先が見える人

プロシージャルの歴史：AI編



プロシージャルAIにおけるいろいろなアルゴリズム

(3) 新しい技術領野を拓くエンジニアリングの必要性



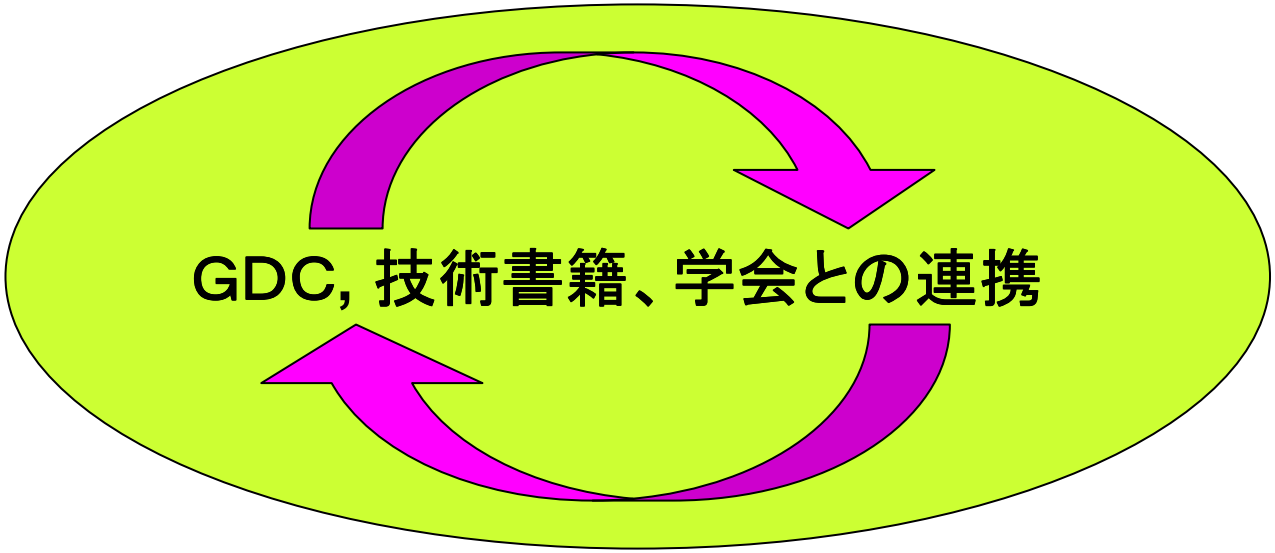
日本の現場はプログラマー文化、エンジニアが生きにくい現場。

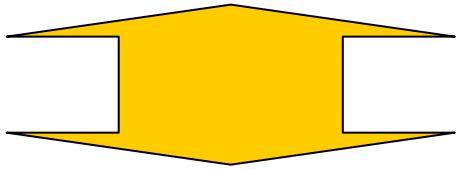
次世代機のスタートで、エンジニアの切り開く力が必要だった。

エンジニアリングの文化を育まなければならない。

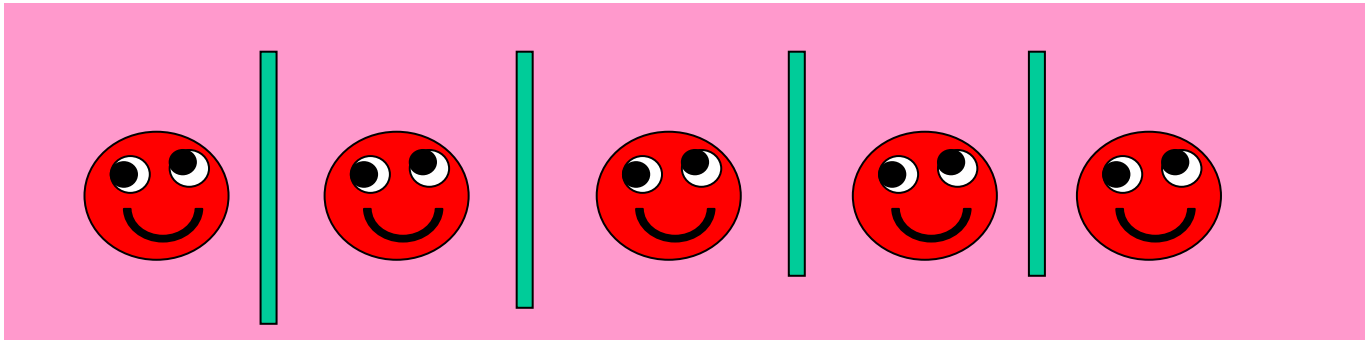
(4) 情報環境の整備

欧米



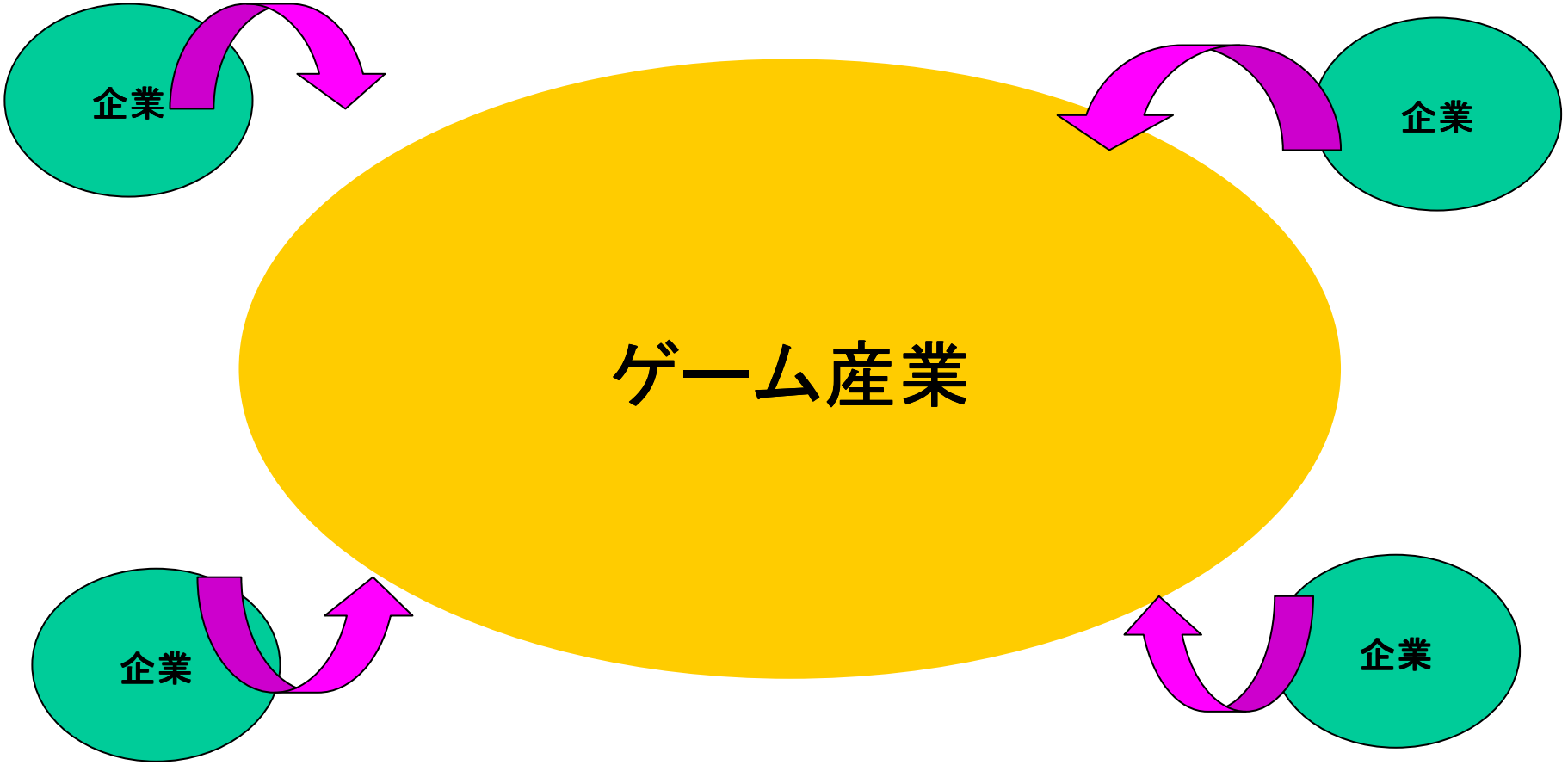


日本



情報マネージメント

こういう問題を考える:
持っている技術情報10のうち、幾つを全体に還元し合うことで、
ゲーム産業全体の利益となり、自社に還元できるか？



改善策

①デジタルゲームをエンジニアリングする姿勢を習慣づける

②エンジニアがゲーム業界に入って来るような体制を作る。

③エンジニアを育てよう。

④情報環境を整備する。情報マネージメントの重要性。

⑤技術を育てる分野を意識する。