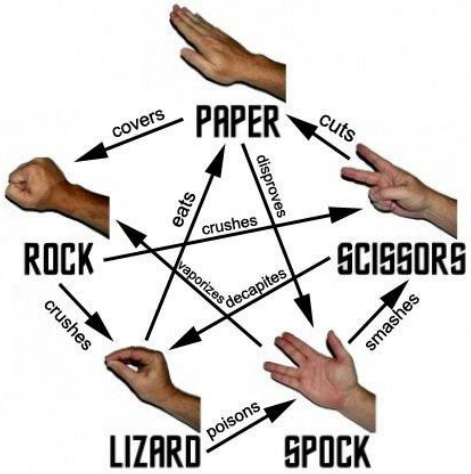
IC-UNICAMP

MC 613



IC/Unicamp

Prof Guido Araújo

Prof Mario Côrtes

Exemplo de projeto hierárquico e modular

Jogo
Rock Paper Scissors Lizard Spock

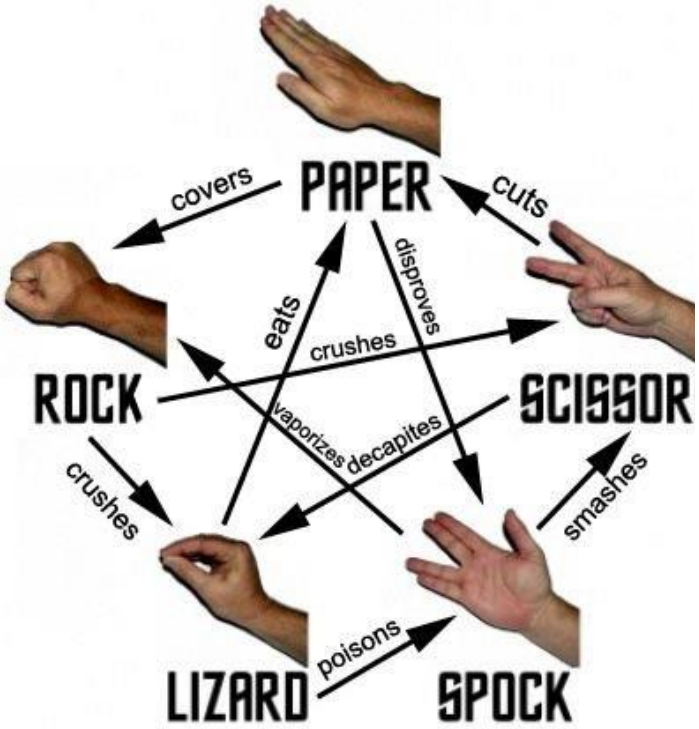
Objetivo do projeto exemplo

- Ilustrar
  - Decomposição do projeto em módulos
  - Relacionamento entre módulos
  - Isolamento de componentes
    - Game engine
    - Interfaces de Entrada e Saída
  - Possibilidade de troca de módulos, por ex:
    - interface inicial com botões, leds e display 7seg
    - interface de entrada envolvendo mouse e teclado
    - interface de saída com monitor
- Preparar alunos para entrega do diagrama de blocos

Regras do jogo exemplo

IC-UNICAMP

- Jogo Rock Paper Scissors Lizard Spock (RPSLS)
  - Variante do Jaquempo: http://pt.wikipedia.org/wiki/Pedra,_papel_e_tesoura
  - regras em http://en.wikipedia.org/wiki/Rock-paper-scissors-lizard-Spock



	Rock	Paper	Scissors	Lizard	Spock
Rock	0, 0	-1, 1	1, -1	1, -1	-1, 1
Paper	1, -1	0, 0	-1, 1	-1, 1	1, -1
Scissors	-1, 1	1, -1	0, 0	1, -1	-1, 1
Lizard	-1, 1	1, -1	-1, 1	0, 0	1, -1
Spock	1, -1	-1, 1	1, -1	-1, 1	0, 0

Exemplo:
Paper ganha de Rock e Spock
Paper perde de Scissors e Lizard

IC-UNICAMP

Especificação do projeto

- Jogo entre computador (J0), jogadores 1 e 2 (J1, J2)
- Entradas:
  - SW[4..0]: 1-out-of-n: Rock Paper Scissors Lizard Spock
  - KEY3: jogada feita
  - KEY0: Reset
- Saídas:
  - HEX3: nº de jogadas
  - HEX[2..0]: circular a cada 1s nos seguintes valores
    - jogadas: código de 0 a 4
    - score da última jogada
      - por oponente → 0 = perde; 1 = empata; 2 = ganha (total de 0 a 4)
    - score acumulado
  - LEDG1 → Espera jogada do J1
  - LEDG2 → Espera jogada do J2

Diagrama de blocos

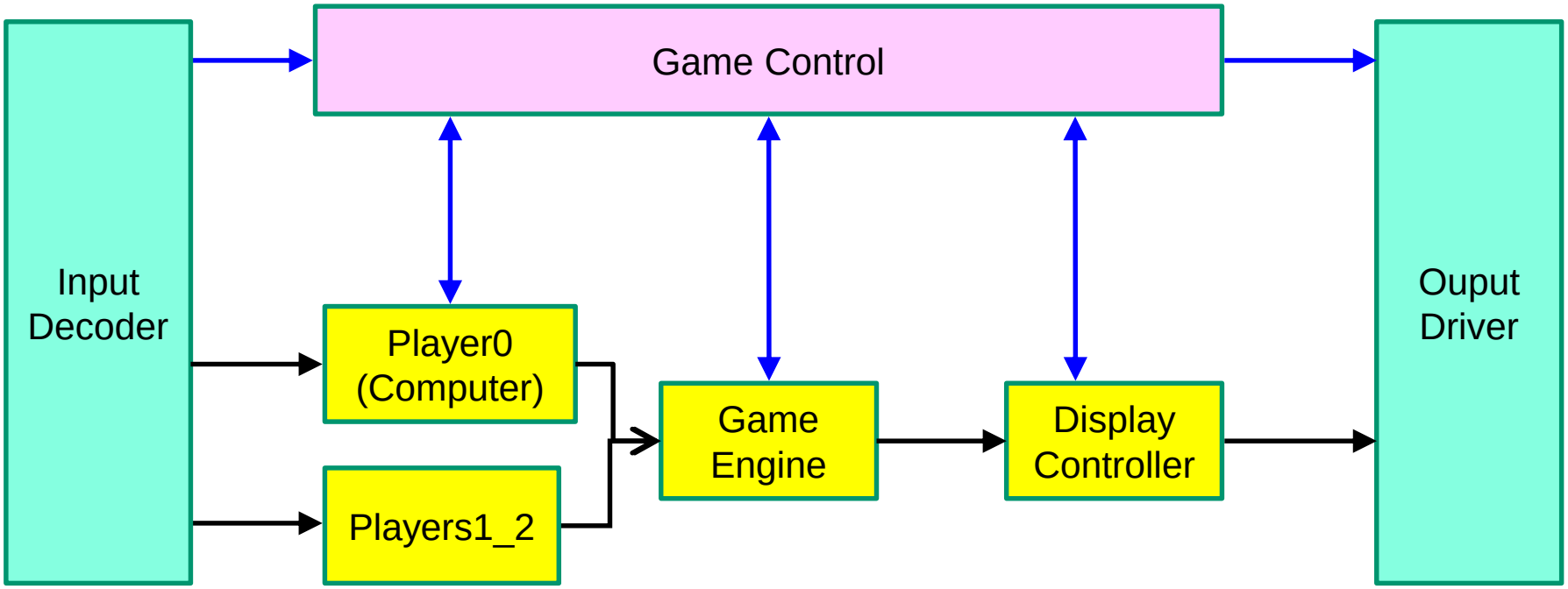
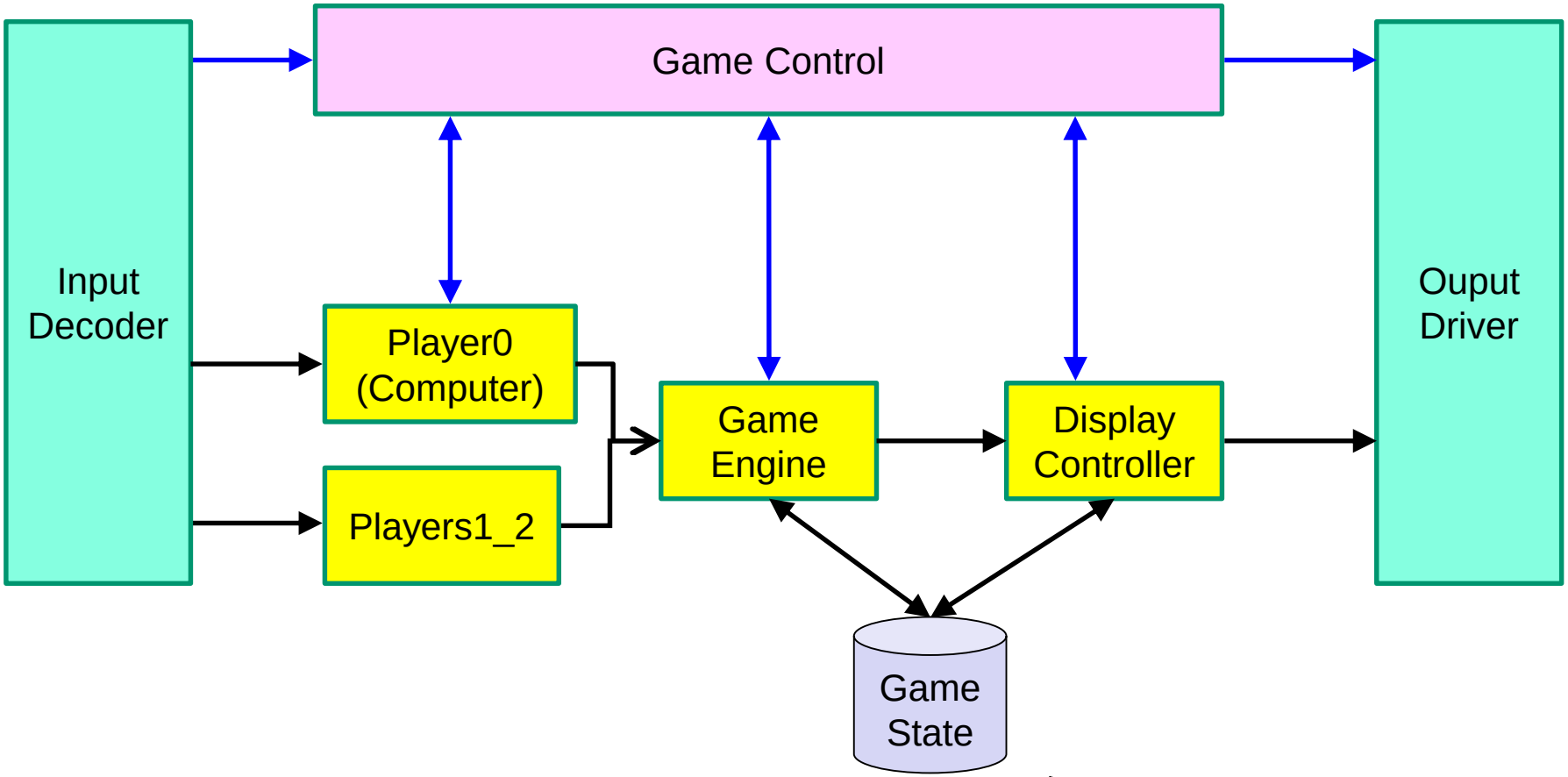


Diagrama de blocos (alternativa)



É comum ter uma estrutura de dados representando a situação instantânea do jogo, interagindo com o Engine e o display. Por exemplo, a posição das peças em um tabuleiro.

IC-UNICAMP

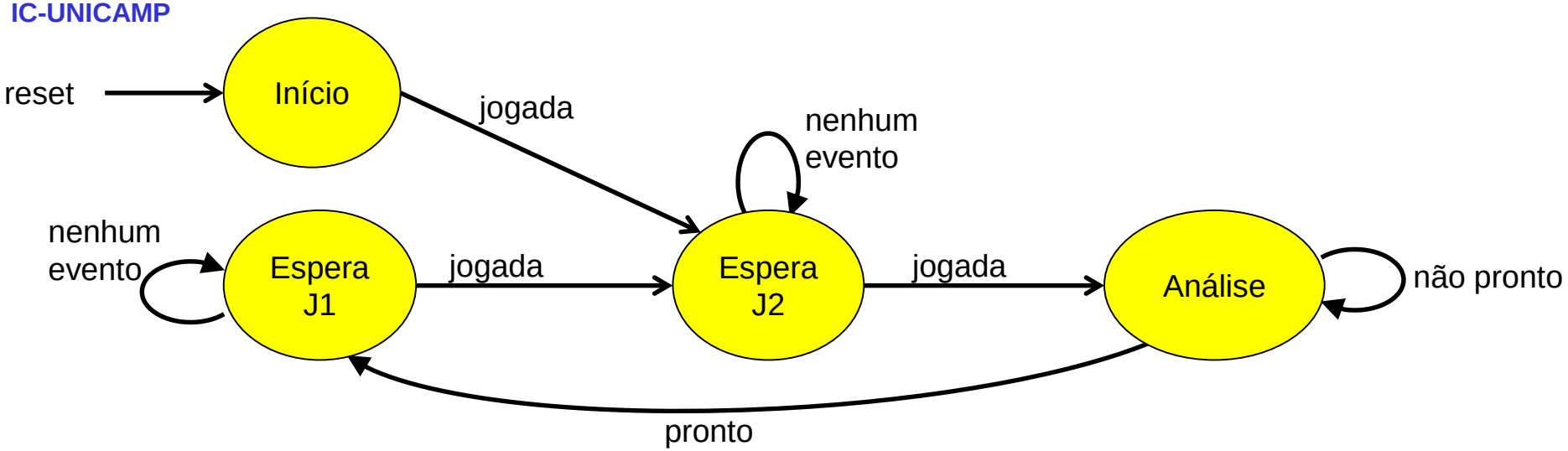
Game State: estrutura de dados

- Estrutura de dados para estado atual do jogo:
  - Representa logicamente o estado atual, por exemplo
    - Situação das peças em um tabuleiro
    - Situação do worm no tabuleiro
  - Deve facilitar a atualização, por ex
    - No worm, decidir o que acontece na próxima jogada e atualizar o estado
  - Interage fortemente: Game Engine, Display

- OBS: aplicar a lógica do jogo sobre a representação lógica e não gráfica

FSM: Game Control

IC-UNICAMP



- LEDG1 espera J1
- HEX indicam score/resultado anterior, ou 0- - - se início

- LEDG2 espera J2
- HEX indicam score/resultado anterior

- Indeterminado, pois a permanência neste estado é curta

IC-UNICAMP

Especificação dos componentes

	Função	Entradas	Saídas
Input Decoder	Tratar entradas	(Jogador)	SW[4..0], KEY3
Players1_2	Converter entradas nos códigos	SW[4..0], KEY3	J12[2..0], jogada
Player0	Gerar jogada aleatória do computador	KEY3	J0[2..0]
Game Engine	Receber jogadas, analisar resultado atual, acumular score	J0[2..0], J1[2..0], J2[2..0], jogada	J0, J1, J2, Score corrente, score acumulado
Display controller	Receber resultados e gerar saídas (inclusive 7 segmentos), temporizar	J0, J1, J2, Score corrente, score acumulado	HEX1, HEX0, LEDG0, LEDR[2..0]
Output driver	(nesta versão tem pouca função: teria mais se saída → monitor)		
Game control	Controlar sequência de estados	Reset, KEY3, pronto	Controle do display, Controle do Game Engine